

AD-A072 427

NAVAL AIR DEVELOPMENT CENTER WARMINSTER PA COMMUNICA--ETC F/G 5/2  
BASIC LABORATORY ARCHITECTURE PLAN.(U)

MAY 79 S GREENBERG, C JOECKEL, R MEJZAK

IDWA-45-78-04

UNCLASSIFIED

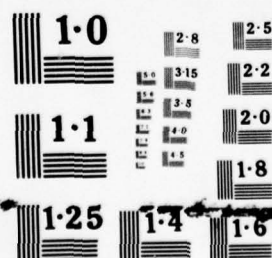
NADC-79161-40

NL

1 OF 4  
AD  
A072427



1 OF 4  
AD  
A072427



NATIONAL BUREAU OF STANDARDS  
MICROCOPY RESOLUTION TEST CHART



REPORT NO. NADC-79161-40

LEVEL

12  
54



A 072427

## BASIC LABORATORY ARCHITECTURE PLAN

Stanley Greenberg, Carl Joeckel and Richard Mejzak  
Communication Navigation Technology Directorate  
NAVAL AIR DEVELOPMENT CENTER  
Warminster, Pennsylvania 18974

17 May 1979

DDC  
RECEIVED  
AUG 7 1979  
RECEIVED  
C

FINAL REPORT  
AIRTASK NO. 6272IN/F21200/WF21200000  
Work Unit No. ZC909

DDC FILE COPY

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Prepared for  
NAVAL AIR SYSTEMS COMMAND  
Department of the Navy  
Washington, D.C. 20361

79 08 06 133

## NOTICES

REPORT NUMBERING SYSTEM - The numbering of technical project reports issued by the Naval Air Development Center is arranged for specific identification purposes. Each number consists of the Center acronym, the calendar year in which the number was assigned, the sequence number of the report within the specific calendar year, and the official 2-digit correspondence code of the Command Office or the Functional Directorate responsible for the report. For example: Report No. NADC-78015-20 indicates the fifteenth Center report for the year 1978, and prepared by the Systems Directorate. The numerical codes are as follows:

CODE	OFFICE OR DIRECTORATE
00	Commander, Naval Air Development Center
01	Technical Director, Naval Air Development Center
02	Comptroller
10	Directorate Command Projects
20	Systems Directorate
30	Sensors & Avionics Technology Directorate
40	Communication & Navigation Technology Directorate
50	Software Computer Directorate
60	Aircraft & Crew Systems Technology Directorate
70	Planning Assessment Resources
80	Engineering Support Group

PRODUCT ENDORSEMENT - The discussion or instructions concerning commercial products herein do not constitute an endorsement by the Government nor do they convey or imply the license or right to use such products.

APPROVED BY:

William F. Lyons Jr.

DATE:

5 July 1979

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 14 NADC-79161-40	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) 6 BASIC LABORATORY ARCHITECTURE PLAN		5. TYPE OF REPORT & PERIOD COVERED 9 Final report
7. AUTHOR(s) 10 Stanley/Greenberg, Carl/Joeckel Richard/Mejzak		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Systems Technology Division (Code 502) Naval Air Development Center Warminster, Pa. 18974		8. CONTRACT OR GRANT NUMBER(s) 15 IDWA-45-78-44
11. CONTROLLING OFFICE NAME AND ADDRESS NADC Communication Navigation Technology Directorate BASIC Laboratory (Code 4092) Warminster, Pa. 18974		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS AIRTASK 6272IN/F21200/ WF21200000; Work Unit ZC909
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Naval Air Systems Command Tech. Admin. AIR-360 Washington, D.C. 20361 12 3027		12. REPORT DATE 17 May 1979 16
		13. NUMBER OF PAGES 103
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
15. DISTRIBUTION STATEMENT (of this Report) Approved for Public Release; Distribution Unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) DDG AUG 7 1979 RECEIVED		
16. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Architecture Plan Multiplexing Laboratory Processing		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Provided herein is a definition of an avionic information processing system architecture and recommendations for implementing a subset of this architecture in the BASIC (Basic Avionic Subsystem Integration Concept) Laboratory. This recommended architecture is classified as a distributed, multiple processing system. It was derived based on the S-3A operational requirements employing a structured system design methodology. The AYK-14 airborne computer and 1553 bus were		

DD FORM 1473

1 JAN 73

EDITION OF 1 NOV 66 IS OBSOLETE  
S/N 0102-LF-014-6601

Unclassified

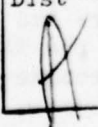
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20.

utilized to the greatest possible extent. A structured system design methodology was employed as a tool in arriving at the architecture in this report.

Accession For	
NTIS GLOWI	<input checked="checked" type="checkbox"/>
DDC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Dist _____	
Approved _____	
Dist	OR
	

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)



## TABLE OF CONTENTS

<u>Section</u>		<u>Page</u>
1	INTRODUCTION .....	1-1
	1.1 Purpose .....	1-1
	1.2 Approach .....	1-1
	1.3 Summary .....	1-2
2	SYSTEM DESIGN METHODOLOGY .....	2-1
	2.1 Methodology Description .....	2-1
	2.2 Application of Methodology to BASIC .....	2-3
3	REQUIREMENTS ANALYSIS .....	3-1
	3.1 Existing S-3A Information Processing	
	System Architecture .....	3-1
	3.1.1 Hardware Configuration .....	3-1
	3.1.2 Operational Program .....	3-6
	3.1.3 Interfaces .....	3-10
	3.1.4 Executive Functions .....	3-11
	3.2 Application Requirements .....	3-14
	3.2.1 Operational Requirements .....	3-15
	3.2.2 Control Requirements .....	3-29
4	APPLICATION STRUCTURE .....	4-1
	4.1 Baseline Generic Structure .....	4-1
	4.2 Assessment Criteria .....	4-3
	4.2.1 Flexibility .....	4-4
	4.2.2 Fault Tolerance/Recovery .....	4-9
	4.2.3 Recovery Techniques .....	4-14
	4.2.4 Software Considerations .....	4-20
	4.2.5 Compatibility with Hardware/Software	
	Standards .....	4-23
5	APPLICATION BINDING .....	5-1
	5.1 Introduction .....	5-1
	5.2 Decompositional Unit Implementations .....	5-2
	5.2.1 Flexibility .....	5-2
	5.2.2 Fault Tolerance/Recovery .....	5-2
	5.2.3 Software Considerations .....	5-2
	5.2.4 Compatibility With Software/Hardware	
	Standards .....	5-3
	5.3 Decompositional Unit Functional Grouping .....	5-3
	5.4 Decompositional Unit/Subsystem Information Flow ....	5-7

## TABLE OF CONTENTS (Continued)

<u>Section</u>	<u>Page</u>
5.5 DU/DU Communication .....	5-7
5.5.1 Modes of Communication .....	5-7
5.5.2 Address Translation Mechanism .....	5-8
5.6 Data/Message Mapping .....	5-8
5.6.1 Data Transfer Rates .....	5-10
5.6.2 MIL-STD-1553A Bus Considerations .....	5-12
5.7 System Configuration Analysis .....	5-14
5.7.1 Data Transfer Analysis .....	5-14
5.7.2 Processing Analysis .....	5-26
6 IMPLEMENTATION SUBSET .....	6-1
6.1 Objective .....	6-1
6.2 Approach .....	6-1
6.3 Subset Configuration .....	6-4
6.4 Scenario .....	6-6
6.4.1 Objective .....	6-6
6.4.2 Description .....	6-6
6.5 System Data .....	6-6
6.5.1 Objective .....	6-6
6.5.2 Static Data .....	6-7
6.5.3 Dynamic Data .....	6-8
6.6 Functional Description of Nodes .....	6-10
6.6.1 Node 70 .....	6-10
6.6.2 Node 82 .....	6-33
6.6.3 Node 30 .....	6-64
6.6.4 Node 90 .....	6-82
6.6.5 Node 21 .....	6-88

## LIST OF ILLUSTRATIONS

<u>Figure</u>		<u>Page</u>
3-1	S-3A Avionic Hardware Configuration .....	3-2
3-2	Functional Flow Diagram Example (Navigation Data Processing) .....	3-17
3-3	Executive Control Functions .....	3-35
3-4	Initialization Control Function Hierarchy .....	3-37
3-5	Recovery Control Function Hierarchy .....	3-39
3-6	System Configuration/Status Control Function Hierarchy .....	3-39
3-7	Bus Control Function Hierarchy .....	3-41
3-8	File/Data Base Management Control Function Hierarchy .....	3-42
4-1	Generic Application Function Architecture .....	4-2
4-2	Software Size/Cost .....	4-22
5-1	Decompositional Unit Information Flow Diagram .....	5-5
5-2	Communication Scheme .....	5-9
5-3	S-3A Data Transfers (Transfer Rates Normalized to Bits per Second) .....	5-11
5-4	MIL-STD-1553A Formats .....	5-13
5-5	MIL-STD-1553 Bus Message Transfer Times .....	5-15
5-6	MIL-STD-1553 Bus Efficiency .....	5-16
5-7	BASIC Architecture .....	5-17
5-8	Message Scheduling .....	5-20
5-9	MIL-STD-1553 Bus Utilization for Synchronous Messages .....	5-25
5-10	Secondary Bus Interfaces .....	5-27
6-1	Implementation Subset .....	6-2
6-2	Data Files .....	6-9
6-3 to		
6-8	Node 70 Executive Functions .....	6-18
6-9 to		
6-23	Node 70 Operational Functions .....	6-23
6-24 to		
6-25	Node 70 Test and Recovery Function .....	6-34
6-26 to		
6-31	Node 82 Executive Functions .....	6-41
6-32 to		
6-60	Node 81 Operational Functions .....	6-46
6-61	Node 81 Test and Recovery Function .....	6-63

## LIST OF ILLUSTRATIONS (Continued)

<u>Figure</u>		<u>Page</u>
6-62 to 6-67	Node 30 Input Processing Function . . . . .	6-67
6-68 to 6-72	Node 30 MAD Processing Function . . . . .	6-70
6-73	Node 30 Controller Monitor Function . . . . .	6-79
6-74	Node 30 Reconfiguration Function . . . . .	6-80
6-75	Node 30 Self-Test Function . . . . .	6-81
6-76 to 6-78	Node 90 Test and Reconfiguration Function . . . . .	6-84
6-79 to 6-83	Node 21 Executive Functions . . . . .	6-91
6-84 to 6-88	Node 21 Operational Functions . . . . .	6-95
6-89 to 6-90	Node 21 Test and Reconfiguration Function . . . . .	6-101



NADC-79161-40  
LIST OF TABLES

<u>Table</u>		<u>Page</u>
3-1	S-3A Avionic Sensors/Subsystems .....	3-4
3-2	Subprogram Nominal I/O Rates .....	3-12
3-3	Operator-Initiated Functions Example (Display) .....	3-18
3-4	Decompositional Units .....	3-21
5-1	Proposed DU Grouping .....	5-4
5-2	Revised DU Grouping .....	5-18
5-3	Message Structure .....	5-22
5-4	Peak/Average Bus Utilization .....	5-26
5-5	Subprogram System Resource Requirements .....	5-29
5-6	Processing Node Sizing Estimates for BASIC Architecture .....	5-29
5-7	Node 10 Configuration .....	5-31
5-8	Configuration for Nodes 21 and 22 .....	5-32
5-9	Node 30 Configuration .....	5-33
5-10	Nodes 40 and 50 Configuration .....	5-34
5-11	Node 70 Configuration .....	5-36
5-12	Nodes 81 and 82 Configuration .....	5-36
5-13	Node 90 Configuration .....	5-37
5-14	Node Hardware Configuration Summary .....	5-38
5-15	Microcomputer Simulator Configuration .....	5-39
5-16	Format of Primitive Operators .....	5-43
5-17	DU - Message Relationships .....	5-47
5-18	AN/AYK-14 Hardware BITE .....	5-51
6-1	Node 70 Input Messages .....	6-11
6-2	Node 70 Output Messages .....	6-12
6-3	Node 70 Command Inputs (RCV SUB ADR 30) .....	6-13
6-4	Node 70 Asynchronous Outputs (XMT SUB ADR 30) .....	6-14
6-5	Node 70 Input Message Formats .....	6-15
6-6	Node 70 Output Message Formats .....	6-16
6-7	Self-Test Message Format .....	6-33
6-8	Node 82 Command Inputs .....	6-36
6-9	Node 82 Input Messages .....	6-38
6-10	Node 82 Output Messages .....	6-39
6-11	Node 82 Output Messages to Display (Node 81) .....	6-40
6-12	Node 82 Data Files .....	6-45
6-13	Node 30 Input Messages .....	6-65
6-14	Node 30 Output Messages .....	6-65
6-15	Node 90 Input Messages .....	6-83
6-16	Node 90 Output Messages .....	6-83
6-17	Node 21 Input Messages .....	6-89
6-18	Node 21 Output Messages .....	6-90

NADC-79161-40  
ACRONYMS

AACS	Aircraft-Altimeter Computer Set
AAES	Advanced Aircraft Electrical System
ACCON	Acoustic Control Subprogram
ACTIVE	Active Acoustic Control Subprogram
ADP	Acoustic Data Processor
AHRS	Altitude and Heading Reference Subsystem
AIDS	Advanced Integrated Display System
AME	Angle-Measuring Equipment
ARMCON	Armament Control Subprogram
ARMCOS	Armament Control Subsystem
ARU	Auxiliary Readout Unit
ASMAT	Active Sonobuoy-to-MAD Attack Tactics
ASW	Antisubmarine Warfare
ATR	Analog Tape Recorder
BASIC	Basic Avionic Subsystem Integration Concept
BEM	Bus Extender Module
BIT	Built-In Test
BITE	Built-In Test Equipment
BT	Bathythermograph
CAINS	Carrier Aircraft Aligned Inertial Navigation System
CASS	Command Activated Sonobuoy System
CCU	Computer Control Unit
COMM	Communication Subsystem
COMPAC	Communication Processing and Control Subprogram
CP	Central Processor
CPU	Central Processing Unit
CRT	Cathode Ray Tube

NADC-79161-40  
ACRONYMS (Continued)

CVS	ASW Support Aircraft Carrier
DAIS	Digital Avionics Information System
DE	Decompositional Element
DEP	Data Extraction Subprogram
DGVS	Doppler Ground Velocity Subsystem
DICASS	Directional CASS
DIFAR	Directional LOFAR
DIM	Discrete Interface Module
DISPAC	Display Processing and Control Subprogram
DME	Direction-Measuring Equipment
DMTU	Digital Magnetic Tape Unit
DTS	Data Terminal Set
DU	Decompositional Unit
EAU	Extended Arithmetic Unit
ESM	Electronic Surveillance Measurement Subprogram or Subsystem
ESR	Executive Service Request
EXEC	Executive Subprogram
FLIR	Forward-Looking Infrared Subsystem or Subprogram
FTP	Fly-to Point
GCSS	Generalized Computer System Simulator
GPM	General Processor Module
GPS	Global Positioning System
HLL	High-Level Language
HOL	Higher Order Language
IDWA	Interdepartmental Work Agreement
IFF	Identification Friend or Foe
IFPM	In-Flight Performance Monitoring Subprogram

NADC-79161-40  
ACRONYMS (Continued)

IFTP	In-Flight Training Program
IHX	Interrupt Handler
ISA	Integrated Inertial Sensor Assembly
IM	Interface Module
INCOS	Integrated Control Subsystem
INS	Inertial Navigation Subsystem
I/O	Input/Output
IOC	I/O Controller
IOP	I/O Processor
JTIDS	Joint Tactical Integrated Data System
KEYPAC	Keyset Processing and Control Subprogram
kops	Thousand operations per second
LOFAR	Low-Frequency Acquisition and Ranging
MAD	Magnetic Anomaly Detection
MCM	Memory Control Module
MPD	Multipurpose Display
MTBF	Mean Time Between Failures
NAV	Navigation Subprogram
NAVAIRDEVCE	Naval Air Development Center
NDRC	Navigation Data Repeater and Converter
OTPI	On-Top Position Indicator
PAC	Passive Acoustic Classification Subprogram
RAAWS	Radar Altimeter and Altitude Warning Subsystem
RADAR	Radar Subsystem or Radar Processing and Control Subprogram
RO	Range-Only
SENSO	Sensor Operator



NADC-79161-40  
ACRONYMS (Continued)

SESCON	Search Stores Control Subprogram
SESCOS	Search Stores Control Subsystem
SIF	Selective Identification Feature
SIM	Serial Interface Module
SLTA	Search, Localization, Track, and Attack
SRS	Sonobuoy Reference Subsystem
SRX	Sonobuoy Receiver Subsystem
SUS	Sound Underwater Signal
TACCO	Tactical Coordinator
TAC/NAV	Tactical/Navigation
TIES	Transmission and Information Exchange System
T/R	Transmit/Receive
VLSI	Very Large-Scale Integration

SECTION 1

INTRODUCTION

1.1 PURPOSE

This report contains the results of the processing system architecture task performed by 5021 under the Basic Avionic Subsystem Integration Concept (BASIC) Architecture Interdepartmental Work Agreement (IDWA) No. 45-78-04. Provided herein is a definition of an avionic information processing system architecture and recommendations for implementing a subset of this architecture in the BASIC laboratory. This recommended architecture can be classified as a distributed, multiple processing system. It was derived based on the S-3A operational requirements employing a structured system design methodology. The S-3A architecture was chosen as a baseline since it typifies avionic information processing requirements. In addition, current Navy standards, such as the AYK-14 airborne computer and 1553 bus, were employed to the greatest possible extent.

One of the major objectives was to provide flexibility so that the architecture can adapt to varying configurations as dictated by new or changing technologies. Another major objective was to provide fault tolerance so that the system can recover from and circumvent failures. These objectives were included in the criteria for assessing the recommended architecture. The modularity aspects of the architecture permit the system to be implemented and verified in an incremental manner. This is a major benefit made possible by structured design.

1.2 APPROACH

A structured system design methodology was employed as a tool in arriving at the architecture in this report. The general procedure followed includes performing a requirements analysis, defining an application structure, establishing

assessment criteria, and mapping the application to hardware and software requirements. The contents of this report, therefore, detail the application of this procedure to the S-3A requirements and other rationale employed in deriving a proposed avionic information processing system architecture.

### 1.3 SUMMARY

This report initially describes the steps involved in the system design methodology (Section 2). Application requirements are derived in Section 3 by examining the existing S-3A processing system. This results in a functional partitioning of the operational program into decompositional units. Section 4 establishes a baseline generic structure as well as assessment criteria relative to flexibility, fault tolerance, and recovery. Implementations for the decompositional units are developed in Section 5. An analysis is performed and rationale is provided for the definition of a proposed architecture. An implementation subset, recommended for the BASIC laboratory, is then described in Section 6, based on demonstrable assessment criteria as previously established.

SECTION 2

SYSTEM DESIGN METHODOLOGY

This section describes the methodology used in development of the recommended architecture for the BASIC laboratory.

2.1 METHODOLOGY DESCRIPTION

The methodology used to derive the proposed system is based upon the Avionic Information Processing System Design Methodology currently being developed by Univac under NAVAIRDEVCON 5021 direction. This methodology is still in a preliminary, qualitative state, but quantifying enhancements will soon be made. A full description can be found in Reference 1.

The methodology of designing information processing systems should be a structured process consisting of the following stages:

- a. Determination of the application requirements
- b. Decomposition of the requirements into decompositional elements (DE's)
- c. Association of the DE's into decompositional units (DU's)
- d. Synthesis of a control structure relating DU's to one another
- e. Implementation selection for each DU in hardware, software, or firmware
- f. Selection of hardware components and software language(s)
- g. Composition of the hardware system
- h. Mapping of software and firmware DU's onto the hardware
- i. Simulation of the system, both hardware and software

In each stage, there is a decision matrix which has columns headed by design choices available to the designer and rows listing assessment criteria appropriate to that design stage. The elements of the matrix are either +, 0, or -, representing a positive choice, neutral choice, or negative choice, respectively. Frequently, qualitative comments also accompany the matrix elements.



The designer adds up the elements in each column (that is, design choice) and chooses the column with the largest net positive total. As stated before, this approach is currently qualitative; but work on quantifying the methodology, that is, assigning numerical weights to the elements, will begin shortly.

The design process will not be examined in more detail. The first stage is the determination of the application requirements. This consists of the following steps: (1) determination of the algorithms that are to be done by each sensor and subsystem; (2) development of the repetition rates and the precision of each algorithm; (3) definition of the information to be received from and sent to each sensor, display, subsystem, and peripheral; and (4) the specification of the required data rates for such transmissions.

The next two steps, application decomposition and association into DU's, involve the partitioning of the requirements into the smallest practical elements (DE's), which are then combined to form the DU's. The forming of DU's makes use of the natural association of certain DE's on the basis of shared resources or data transmission between the grouped elements.

The fourth stage, synthesis of a control structure, involves determination of the control dependencies among DU's and the relative priorities of the DU's. It should be noted here that, thus far in the design procedure, no assumption has been made concerning the implementation of the DU's. Whether they will be implemented in hardware, software, or firmware is of no importance in the first four steps. However, in the fifth stage, each DU is bound to a specific medium of implementation.

The sixth and seventh stages are self-explanatory. At this time, the hardware components are selected and configured into a system. The hardware can consist of both dedicated hardware logic for those DU's implemented in hardware and processors, memories, buses, and input/output (I/O) peripherals

for performing the software- and firmware-implemented DU's. The selection of software language(s) and support software also occurs during the sixth stage.

In the eighth stage, the software and firmware DU's are mapped onto the hardware chosen previously. This mapping can be either static, dynamic, or a hybrid of the two. An example of the latter might be an initial static allocation of tasks to various processors in a distributed system plus a dynamic reconfiguration capability in case of a fault in a processor. Alternatively, the dynamic mapping algorithms can also be considered as another DU and could be handled in the earlier stages. This issue is still to be decided in the final methodology.

In the ninth stage, the entire system, both hardware and software, is simulated. During this stage, tradeoff studies can be made, the system can be "fine-tuned," and the final system validated. Should the simulation studies show that modifications must be made in the system, the methodology procedure can be re-entered at any of the earlier stages, the changes made, and the succeeding steps repeated with the new inputs.

## 2.2 APPLICATION OF METHODOLOGY TO BASIC

The application requirements for the BASIC laboratory were assumed to be functionally equivalent to those of the S-3A avionic suite less some of the sensor processing. Obviously, the subsystems to be used in future platforms will differ from those in the S-3A; but, since the information processing requirements for these future subsystems are not clearly defined at present, it was decided to use the S-3A avionic requirements to design an initial baseline system. As the requirements for the new subsystems become available, they can be inserted into the first stage of the methodology, and a new information processing system can be designed. However, if the initial information processing system architecture is flexible enough, it should be able to accommodate the changed requirements.

In applying the methodology to the S-3A requirements for the BASIC design, the first three stages of the methodology were followed in the described procedure to partition the requirements into DU's. However, in the fourth through the eighth stages, the methodology was not fully utilized. There are several reasons for this. First, the methodology is still qualitative and not yet fully developed. Second, it was assumed that all algorithms would be performed in software because of the decision that the BASIC laboratory should not be involved at this time with special-purpose hardware or firmware logic; rather, the laboratory should stress the development of a flexible, expandable system. Third, processors used in naval aircraft for the next decade will almost surely be standard minicomputers (AN/AYK-14 or successors) and possibly standard microprocessors (not yet chosen); so the proposed system architecture is being developed based on these assumptions. Finally, the advances being made in computer technology during the last couple of years virtually lead a designer to choose a distributed/hybrid architecture because of the increasing disparity in hardware costs versus software cost.

The last step in the methodology, that of verification via simulation, has not yet been performed for the recommended architecture, but the simulation inputs are presently being prepared and results will be available in a later supplement to this report.

## REQUIREMENTS ANALYSIS

The first step in the design methodology is the determination of application requirements. This section derives application requirements, in terms of implementation-independent decompositional units, from an extensive examination of the existing S-3A processing system.

## 3.1 EXISTING S-3A INFORMATION PROCESSING SYSTEM ARCHITECTURE

The following paragraphs provide a functional overview of the current implementation of the S-3A information processing system with respect to hardware configuration, operational program, interfaces, and executive functions. Although this information is detailed in various related S-3A documentation, a summary is provided here for reference. This system has been used as a baseline for development of the BASIC architecture.

3.1.1 Hardware Configuration

The S-3A information processing system hardware is based on the AN/AYK-10 general-purpose digital computer. All the antisubmarine warfare (ASW) tactical processing is controlled by this computer. The AN/AYK-10 has two central processors which communicate with two 32-K memory units over instruction and operand buses. In addition, two I/O controllers conduct and direct all I/O traffic between I/O devices and memory. A block diagram of the S-3A digital avionics hardware configuration is shown in Figure 3-1.

3.1.1.1 Central Processor

Each central processor is composed of a control section and an arithmetic section. The control section interprets the instruction, carries out the commands, and directs the sequence of events for the logical execution of avionic programs and external interrupts. The arithmetic section performs the arithmetic and logic functions as directed by the control section. Each processor operates independently, but both use the same instruction repertoire consisting



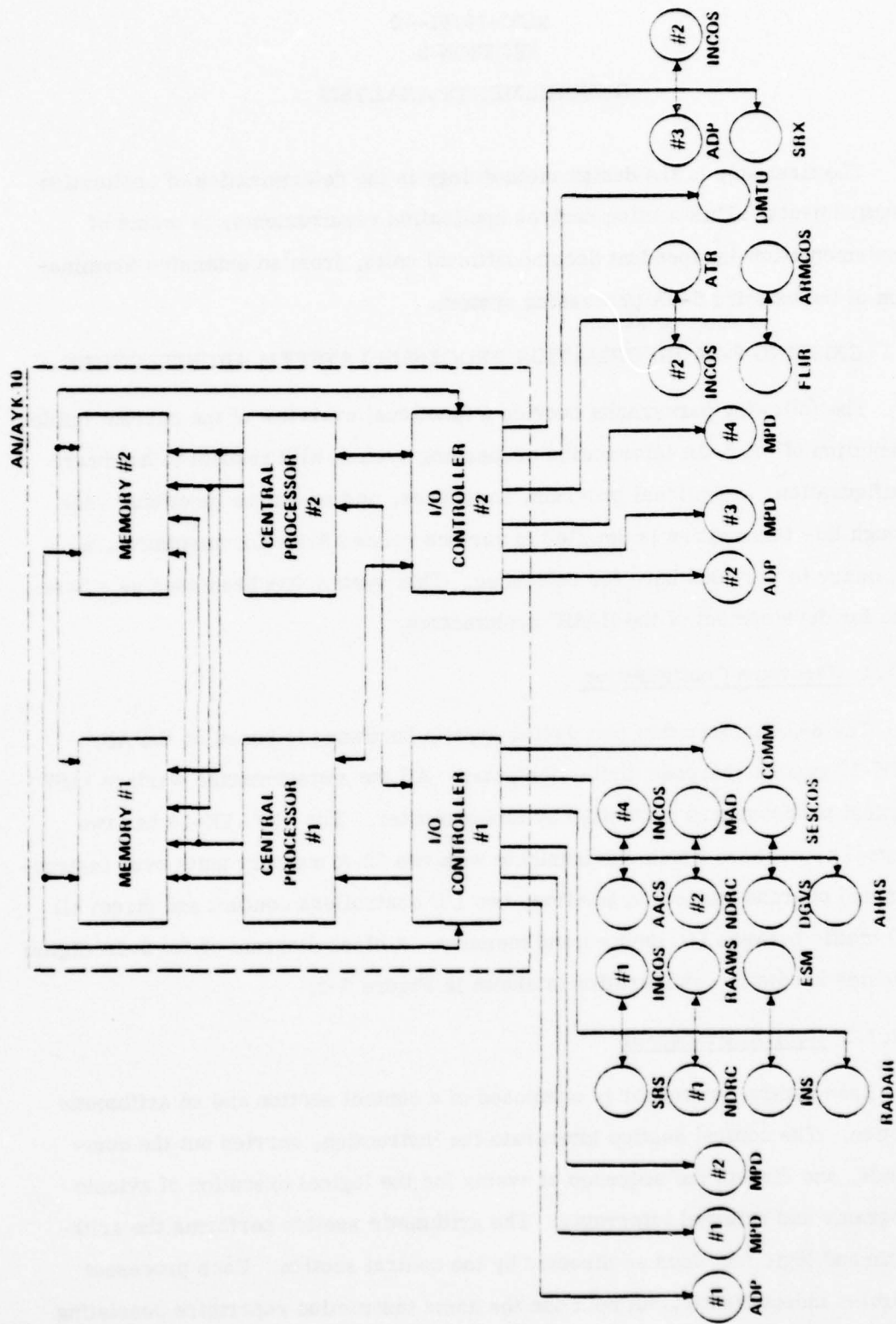


Figure 3-1. S-3A Avionic Hardware Configuration

of arithmetic, logical, compare, and I/O operations. Additional processor speed is attained through the use of memory overlap and a high speed shift matrix. Memory overlap involves retrieving the present operand from one memory and the next instruction from another memory in parallel over two different memory buses.

#### 3.1.1.2 Memory Units

Each of two memory units provides up to 32,718 36-bit words (32 data plus 4 parity). Each memory contains its own addressing circuitry which allows each unit to function independently while servicing six different users. The memory has an average cycle time of 750 nanoseconds.

#### 3.1.1.3 I/O Controller

The I/O controller and interface unit provide the control logic which (1) directs the sequence of events for the logical execution of input and output programs and (2) conducts the transfer of data between I/O devices and memory. The I/O controller requires only a single initiation command from a processor to initiate an I/O command program (chain). This I/O command program uses its own command repertoire in controlling I/O operations, and it has direct memory access through its own I/O memory bus. The AN/AYK-10 provides a serial I/O interface to S-3A avionics in which some channels may have up to seven devices multiplexed together.

#### 3.1.1.4 Avionic Sensors/Subsystems

A list of the avionic devices, their nomenclature, and their function is presented in Table 3-1. Each device is connected to the AN/AYK-10 via a Manchester serial channel. The maximum channel data rate for the acoustic data processors (ADP's) Numbers 1 and 2 is 150 K words per second. The maximum data rate for the remaining devices is 100 K words per second. Normally, these peripherals run at a small percentage of their maximum data rate.

TABLE 3-1. S-3A AVIONIC SENSORS/SUBSYSTEMS

NOMENCLATURE	NAME	FUNCTION
ADP	Acoustic Data Processor	Provides acoustic sensor data.
MPD	Multipurpose Display	Displays sensor and tactical data to operators.
RADAR	Radar Subsystem	Locates objects by receiving transmitted RF energy.
INS	Inertial Navigation Subsystem	Provides aircraft position, velocity, and heading.
NDRC	Navigation Data Repeater and Converter	Interfaces with other navigation subsystems to provide aircraft position, course, speed, altitude, and guidance.
SRS	Sonobuoy Reference Subsystem	Provides sonobuoy position relative to aircraft.
INCOS	Integrated Control Subsystem	Provides data entry for aircraft operators.
RAAWS	Radar Altimeter and Altitude Warning Subsystem	Provides aircraft altitude above terrain.
ESM	Electronic Surveillance Measurement Sub-systems	Provide target information from received RADAR signal.
AACS	Aircraft-Altimeter Computer Set	Provides air speed, temperature, and altitude.
DGVS	Doppler Ground Velocity Subsystem	Provides aircraft velocity.

TABLE 3-1. S-3A AVIONIC SENSORS/SUBSYSTEMS (Continued)

NOMENCLATURE	NAME	FUNCTION
AHRS	Altitude and Heading Reference Subsystem	Provides aircraft roll, pitch, and heading.
MAD	Magnetic Anomaly Detection	Provides target information from detecting anomaly in earth's magnetic fields.
SESCOS	Search Stores Control Subsystem	Provides for the release of search stores (for example, sonobuoys).
COMM	Communication Subsystem	Provides for both voice and tactical communication between forces.
FLIR	Forward-Looking Infrared Subsystem	Provides target information for received infrared radiation.
ATR	Analog Tape Recorder	Provides for recording of audio, acoustic, MAD, and digital data for later playback.
ARMCOS	Armament Control Subsystem	Provides control for weapon launching.
DMTU	Digital Magnetic Tape Unit	Provides for system loading and operational event recording.
SRX	Sonobuoy Receiver Subsystem	Provides for signal data communication from sonobuoys to aircraft.



### 3.1.2 Operational Program

The S-3A operational program is divided into functional areas called subprograms. A brief description of each subprogram is provided in the following paragraphs.

#### 3.1.2.1 Executive Subprogram (EXEC)

The EXEC subprogram provides the initialization, scheduling, input/output control, error processing, and system recovery for the rest of the subprograms. A further discussion regarding the executive functions is provided in paragraph 3.1.4 of this document.

#### 3.1.2.2 Keyset Processing and Control (KEYPAC) Subprogram

The KEYPAC subprogram provides the primary man-machine interface for the operational system. KEYPAC obtains control via the EXEC through interrupts received from the keysets located at the operator stations. Upon receiving control, KEYPAC provides for (1) keyset mode control, (2) matrix select switch control, (3) switch lighting, (4) keyset hardware monitoring, and (5) initiating the operator specified functions.

#### 3.1.2.3 Display Processing and Control (DISPAC) Subprogram

The DISPAC subprogram provides for the visual means of communicating information from the various avionic subsystems to the S-3A crewmember via the multipurpose displays (MPD's). In this function DISPAC will (1) display operator alerts, (2) display and process cues, (3) display and process tableaux, and (4) control tactical symbology.

#### 3.1.2.4 Navigation (NAV) Subprogram

The NAV subprogram provides navigation parameters to other subprograms. These navigation parameters include (1) the aircraft geographic position, (2) the correct relative buoy positions, (3) the carrier position, and (4) other operator navigational aids.

### 3.1.2.5 Steering Subprograms

The steering program provides for automatic steering of the aircraft. The steering subprogram directs the aircraft via an automatic flight control system to fly to or orbit an operator entered fly-to point.

### 3.1.2.6 Search, Localization, Track, and Attack (SLTA) Subprogram

The SLTA subprogram provides a collection of aids which allows the tactical coordinator (TACCO) to correlate tactical information available to him so he can employ the aircraft in the most effective way to accomplish its mission. These aids include (1) standard ASW flight pattern selections, (2) fix and track generation, and (3) target tracking algorithms.

### 3.1.2.7 Communication Processing and Control (COMPAC) Subprogram

The COMPAC subprogram provides communication capabilities with other S-3A's and surface ships. The COMPAC subprogram (1) configures, monitors, and maintains the communication equipment, (2) participates in the operational data link nets, (3) transmits tactical data to other participating units, and (4) passes received tactical data to interested subprograms.

### 3.1.2.8 Ballistic Subprogram

The ballistic subprogram uses weapon or search stores flight characteristics in order to locate drop-and-splash points to within a 100-yard (30.4-meter) radius. The ballistic subprogram provides (1) weapon fly-to points, (2) weapon splash points, and (3) expendable splash points.

### 3.1.2.9 Armament Control (ARMCON) Subprogram

The ARMCON subprogram provides the means to select, arm, and release weapons loaded on wing pylons, triple ejector racks, and bomb bay stations. The ARMCON subprogram (1) enters weapon fly-to points into the system, (2) selects weapons for release, (3) controls the release, and (4) updates weapon inventory.

#### 3.1.2.10 Search Stores Control (SESCON) Subprogram

The SESCOON subprogram provides the ability to select, prepare, designate, and release search stores from sonobuoy launch chutes. The SESCOON subprogram (1) enters search stores fly-to points into the system, (2) selects search stores for release, (3) controls the release, (4) updates the search stores inventory, and (5) allows for designating a sonobuoy currently in the water.

#### 3.1.2.11 Acoustic Control (ACCON) Subprogram

The ACCON subprogram provides the man-machine interface for controlling the acoustic data processor (ADP), the analog tape recorder (ATR), and the sonobuoy receiver set (SRX). The ACCON subprogram (1) maintains control of ADP, ATR, SRX input and output operations, (2) controls the acoustic display modes, (3) controls the audio headsets, (4) assigns sonobuoy receiver frequencies, and (5) manages acoustic contacts.

#### 3.1.2.12 Passive Acoustic Classification (PAC) Subprogram

The PAC subprogram provides a method of classifying passive acoustic signals according to their origin. The PAC subprogram (1) edits digitized acoustic data, (2) alerts operator of active contacts, (3) classifies targets, and (4) manages contact data.

#### 3.1.2.13 Active Acoustic Control (ACTIVE) Subprogram

The ACTIVE subprogram provides target localization by controlling and processing the echoes from a sonic pulse transmitted by an active sonobuoy. The ACTIVE subprogram (1) controls different active modes, (2) controls the sequence of pings, (3) analyzes ping returns, (4) manages active contact, and (5) allows display of targets.

#### 3.1.2.14 Forward-Looking Infrared (FLIR) Subprogram

The FLIR subprogram aids the operator in identifying targets by controlling the FLIR subsystem. The FLIR subprogram (1) controls automatic FLIR

tracking of fixed geographic positions, (2) controls automatic FLIR slewing, (3) provides for entry of FLIR fixes, and (4) allows for adjustment of FLIR field of view, polarity, brightness, and contrast.

#### 3.1.2.15 Magnetic Anomaly Detection (MAD) Subprogram

The MAD subprogram provides the capability to detect, evaluate, and record anomalies in the earth's magnetic field due to the presence of submarines. The MAD subprogram (1) displays MAD sensor data on display, (2) provides for automatic signal detection, and (3) provides for entry of MAD contacts.

#### 3.1.2.16 Radar Processing and Control (RADAR) Subprogram

The RADAR subprogram provides the capability to detect, classify, localize, and track submarine snorkels and surface vessels through the use of the RADAR subsystem. The RADAR subprogram (1) selects the type and format of the radar to be displayed, (2) allows for adjustment of antenna position, power levels, radar mode, persistent video gain, receiver gain, and false alarm rate, (3) provides for entry of RADAR fixes, and (4) provides for auto tracking of a selected target.

#### 3.1.2.17 Electronic Surveillance Measures (ESM) Subprogram

The ESM subprogram provides the capability of detecting, localizing, and classifying intercepted radar transmissions. The ESM subprogram (1) displays selected emitters, (2) provides for the entry of ESM contacts, (3) eliminates redundant emitters, (4) classifies emitters, and (5) controls the scan frequency bands.

#### 3.1.2.18 Data Extraction (DEP) Subprogram

The DEP subprogram provides for the recording of specific tactical events on the airborne digital magnetic tape unit (DMTU).



### 3.1.2.19 System Common Routine Subprogram

The system common routine subprogram is a collection of common routines available for direct use by any task without going through the executive. These routines are categorized as mathematical, conversion, and user routines. One program copy of each common routine is employed for all users.

### 3.1.2.20 In-Flight Performance Monitoring (IFPM) Subprogram

The IFPM subprogram is designed to detect, verify, and take appropriate action in response to error indications from the S-3A avionic equipment units. When the IFPM determines that a subsystem or component of the processing system meets a predefined failure criteria, it communicates this information to the executive, which alerts the operators and indicates the status of the subsystem in a system status tableau.

### 3.1.3 Interfaces

The following paragraphs provide a description of the interfaces which enable the operator, software, and hardware elements of the system to interact.

#### 3.1.3.1 Operator

The S-3A is an operator-oriented system with crew stations consisting of pilot, copilot, tactical coordinator (TACCO), and sensor operator (SENSO). The S-3A aircraft crew communicates with the avionic system through alphanumeric/video multipurpose displays (MPD's) and functionally oriented keyboards.

#### 3.1.3.2 Software

There are three basic ways that elements of the various subprograms interface with one another. The first is at the task level in which one task can schedule another through the executive. The second occurs when a task calls on a subroutine (common routine) to perform some function with parameters passed in the computer registers. The third method is through shared data, the most common being the system common data.

### 3.1.3.3 Hardware

Chains (programs) of I/O controller (IOC) commands direct the I/O controller/ interface to initiate communication between the computer and the various S-3A peripheral subsystems. When a task requires data to be transferred to or from a peripheral device, the following steps are performed:

- a. A data memory area (data buffer) is set up which either contains the data to be transferred or the storage area to which the data is to be sent.
- b. The subprogram calls on the EXEC subprogram to initiate an IOC program (chain).
- c. The EXEC directs the IOC to initiate this specific chain by passing the channel number and chain address to the IOC over the central processing unit (CPU) to IOC data bus.
- d. The IOC initiates this chain, which causes data words to be transferred between the peripheral and data buffer. This data transfer is under complete control of the IOC chain; thus, the CPU is free to do other processing while the data transfer is in process.
- e. Upon completion of the transfer, the IOC chain usually interrupts the CPU and notifies the EXEC of the completion. The EXEC will either notify the requesting subprogram or a subprogram interrupt handler (IH) of this completion.

Most of the I/O operations in the S-3A Operation Program occur at a periodic rate and are requested by periodic tasks with that same rate.

Table 3-2 is a list of each subprogram and the I/O devices it communicates with. For each I/O device, the nominal duty cycle (average time interval between data transfers) and the data transfer rate are listed.

### 3.1.4 Executive Functions

The purpose of the executive is to provide an initial environment for the operational program and to provide a focal point for the processing of all interrupts. Since most of executive processing requires the use of privileged instructions, the majority of the executive operates in the interrupt state. During mission operations, the executive is primarily concerned with the management of space and time resources.

TABLE 3-2. SUBPROGRAM NOMINAL I/O RATES

SUBPROGRAM	I/O DEVICE	NOMINAL DUTY CYCLE	NOMINAL RATES (words per second)
EXEC	ADP No. 1 (Drum 1)	Demand	12000
	ADP No. 2 (Drum 2)	Demand	12000
KEYPAK	INCOS No. 1	50 msec + Demand	80
	INCOS No. 2	50 msec + Demand	80
	INCOS No. 4	Demand	
DISPAC	MPD No. 1	25 msec	20000
	MPD No. 2	25 msec	20000
	MPD No. 3	25 msec	18000
	MPD No. 4	25 msec	6000
NAV	NDRC No. 1	200 msec	60
	NDRC No. 2	200 msec	60
	CAINS	50 msec	120
	RAAWS	200 msec	5
	CSAA	200 msec	10
	DGVS	200 msec	30
	AHRS	50 msec	80
	SRS	1 sec	40
Steering	(None)	-	-
SLTA	(None)	-	-
COMM	COMM	Demand	100
Ballistics	None	-	-
ARMCON	ARMCOS	Demand	24
SESCON	SESCOS	Demand	10
ACCON	ADP No. 1	Demand	700
	ADP No. 2	Demand	700
	ADP No. 3	100 msec (max)	100
	SRS	1 sec	4
	ATR	100	10

TABLE 3-2. SUBPROGRAM NOMINAL I/O RATES (Continued)

SUBPROGRAM	I/O DEVICE	NOMINAL DUTY CYCLE	NOMINAL RATES (words per second)
PAC	(None)	-	-
ACTIVE	(None)	-	-
FLIR	FLIR	50 msec	75
MAD	MAD	42 msec	24
RADAR	RADAR	50 msec (max)	110
ESM	ESM	250 msec	100
DEP	DMTU	1 sec/demand	5000

#### 3.1.4.1 Space Resources

Available areas of the ADP drums and main memory units are initially available for use of nonresident segments called the transient area. The allocation, retrieval, and drum handler portions of the executive manage the allocation of this space to drum segments using mapping tables and queues for requests.

#### 3.1.4.2 Time Resources

Time resources include central processor (CP) and IOC utilization and the allocation of I/O channel time to various requestors. S-3A tasks are the prime users of CP time. The scheduler and dispatcher portions of the executive have directories and queues to assist in controlling CP utilization. The centralized I/O portion of the executive controls the usage of the IOC and its channels.

#### 3.1.4.3 Executive Service Requests

A variety of executive service requests (ESR's) are used by the S-3A tasks to interface with the executive for resource utilization. These services allow a task to schedule other tasks, to read or write drum segments, to request I/O



initiation, to save CP registers, to lock or unlock data tables, to allocate transient space for temporary data storage, to establish or remove wait conditions, and to increase allowable CP execution time.

#### 3.1.4.4 Operational Program Priority Levels

Each function within the S-3A operational program is assigned to one of eight levels of priority. The basic driving force of the executive is the scheduling queue, which contains the list of tasks currently scheduled according to their priority. This table is used by the executive to select the highest priority task and allows the task to use CP time. When tasks are encountered that are drum-resident, the executive allocates memory space to the segment and initiates retrieval from the drum. When the segment retrieval is complete, any scheduled tasks of this segment are eligible for dispatch.

#### 3.1.4.5 Executive Recovery Component

The operational program is capable of accommodating a variety of equipment failures without loss of all capability. Failures in the avionic equipment are addressed in the IFPM subprogram, as indicated in paragraph 3.1.2.20. Failures resulting from a loss of the executive resources of time and space will be handled by the executive recovery component, which is vested with the responsibility of maintaining some meaningful operational capability in the light of such resource failures. Depending upon the nature of the failure, some capability may be lost; but, in general, techniques used by the recovery component will either reinitialize the system or reload via the digital magnetic tape unit (DMTU). However, for all failures, the operator is notified via system alerts; and the error is extracted on the DMTU.

### 3.2 APPLICATION REQUIREMENTS

The following paragraphs discuss the results of applying the first three stages of the methodology to the current S-3A avionic information processing

system. As indicated previously, these stages are concerned with the application requirements and functional decomposition aspects of the methodology and are employed to partition the S-3A information processing system into functional elements establishing baseline requirements for performing decentralized architectural studies. In addition, control strategy requirements for implementing decentralized architectures are also addressed.

### 3.2.1 Operational Requirements

Operational requirements are determined by means of analyzing the functional aspects of the subprograms/tasks which constitute the S-3A operational program (see paragraph 3.1.2) followed by a functional partitioning of these software elements into decompositional units and elements.

#### 3.2.1.1 Operational Program Functional Analysis

The S-3A operational program can essentially be considered to consist of subprograms executing on either a periodic or demand basis, as indicated in Table 3-2. Each high-level subprogram module is made up of a group of low-level executable modules called tasks. A task, which can either continuously run at a periodic rate (periodic task) or run as a result of an external interrupt (demand task) usually has a defined priority and iteration rate. However, a subprogram which is made up of many tasks does not have a definite priority or iteration rate. For a high-level analysis, it can be assumed that the subprogram takes on the priority of the task with the highest priority in the subsystem and the iteration rate of the task with the highest periodic rate.

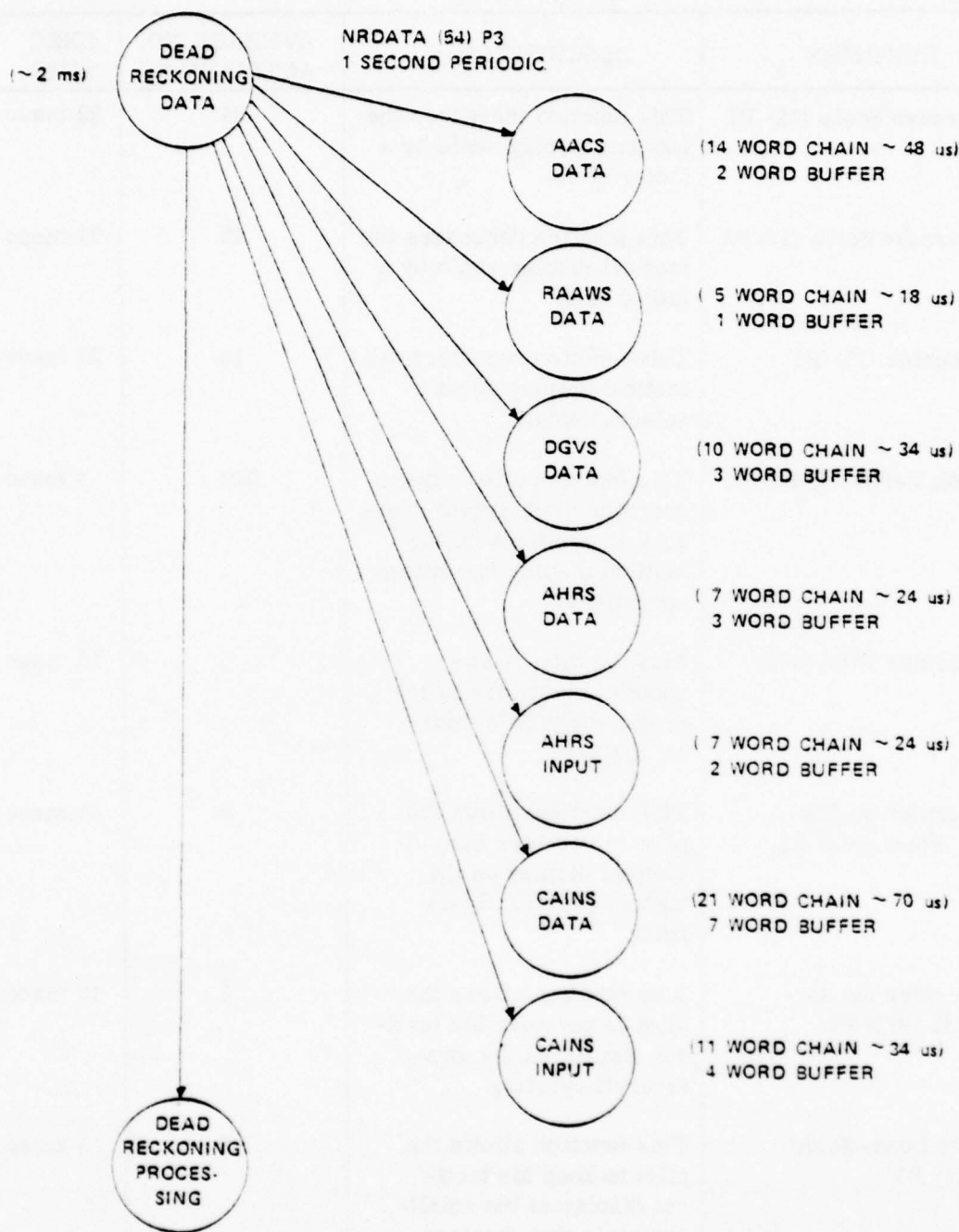
3.2.1.1.1 Functional Flow Diagrams. Each subprogram is divided into many low-level functions, and each function is composed of one or more tasks (executable code). Information, including processing time, priority, and iteration rates, was compiled for these low-level functions and characterized by means of functional flow diagrams. Functional flow diagrams consist of process strings, represented by bubbles, required to perform the function. Due to the extent of these flow diagrams, only the navigation data processing function is provided as

an example in Figure 3-2. The reader is referred to Volume 1, "S-3A Functional Analysis," of the "S-3A Information Processing Architecture Study" for a complete description (Reference 2). As shown in Figure 3-2, the flow diagrams are arranged such that processes performed in the CP are called tasks, shown on the left side, and processes performed in the IOC are called chains, shown on the right side. Bubbles connected by lines indicate that the process represented by the first bubble has scheduled the process represented by the second bubble. Thus, each subsequent bubble is referred to as a successor of the immediately preceding bubble.

3.2.1.1.2 Operator Initiated Functions. Functions that are initiated frequently by S-3A operators are considered as having a major impact on the system and are included in the previously discussed functional flow diagrams. Other operator-initiated functions that only cause momentary increases in system activity are considered as having minor impact on the system. The minor impact functions are summarized in tables, an example of which is shown in Table 3-3 for operator-initiated functions relative to displays. The reader is again referred to the S-3A functional analysis report (Volume 1 of Reference 2) for a complete description of operator-initiated functions.

3.2.1.2 Decompositional Elements/Units Partitioning

An S-3A requirement decomposition report was produced under the S-3A Information Processing Architecture Study contract and contains the rationale and approach for partitioning S-3A functions into decompositional elements (DE's) and decompositional units (DU's) (Reference 3). A DE is defined as "a functional part of a system/application without regard to its execution." A DU is defined as "the basic component of the methodology consisting of one or more decompositional elements, implemented in hardware, software, or firmware and executed as required." As a result of the decompositional analysis, a total of 47 DU's are realized; each DU comprises various DE's. These DU's are



THIS TASK INPUTS NAVIGATION DATA FROM THE NAVIGATION SUBSYSTEM AND PERFORMS DEAD RECKONING PROCESSING TO OBTAIN THE AIRCRAFT POSITION. THIS FUNCTION IS ACTIVE 100% OF THE TIME.

Figure 3-2. Functional Flow Diagram Example  
(Navigation Data Processing)



TABLE 3-3. OPERATOR-INITIATED FUNCTIONS EXAMPLE (DISPLAY)

FUNCTION	DESCRIPTION	AVERAGE NO. ACTIVATIONS	EXEC TIME
Increase Scale (12) P1	This function increases the tactical display scale by a factor of 2.	25	29 msec
Decrease Scale (13) P1	This function decreases the tactical display scale by a factor of 2.	25	21 msec
Recenter (11) P5	This function recenters the tactical display about a selected point.	10	25 msec
Hook Verify (7) P1	This function allows the operator to designate symbols or positions on the tactical display for future operations.	500	8 msec
Recenter Pilot (450) P3	This function allows nonpilot operators to recenter the pilot's tactical display.	2	15 msec
Recenter on Fly-To Point (403) P1	This function allows the pilot to recenter his tactical display on the highest priority fly-to point.	5	15 msec
Recenter On Aircraft (402) P1	This function allows the pilot to recenter his tactical display on the own-aircraft symbol.	5	15 msec
Auto Down-Scale (414) P1	This function allows the pilot to keep his tactical display at the smallest scale that displays the own-aircraft symbol.	2	1 msec
Extend Track Vector (552) P1	This function allows the pilot to extend the track bearing of the own-aircraft symbol to the edge of his tactical display.	2	3 msec

listed in Table 3-4 according to their associated sensors/subsystems, and a summary of the general types of functions performed by each DU is provided in the following paragraphs.

3.2.1.2.1 ESM Intercept Maintenance (DU01)

- Accesses, processes, and updates ESM emitter file
- Places tactical display in ESM display mode
- Displays intercepts on the MPD

3.2.1.2.2 ESM Frequency Control (DU02)

- Performs control functions associated with frequency bands in which the ESM receiver scans
- Permits operator modification of scan bands and inhibits lists

3.2.1.2.3 ESM Classification (DU03)

- Displays possible classifications of designated signal
- Permits the operator to specify classification of specific intercept

3.2.1.2.4 ESM Contact Maintenance (DU04)

- Enters intercept in contact file

3.2.1.2.5 ADP Mode Control (DU05)

- Controls various modes and functions of the ADP
- Performs initialization of ADP subsystem and acoustic functions
- Provides various target information such as target locations relative to sonobuoys

3.2.1.2.6 ADP Data Management (DU06)

- Obtains all acoustic data inputs from the ADP and stores data in a central file for use by other elements

3.2.1.2.7 ADP Display Control (DU07)

- Establishes the MPD and auxiliary readout unit (ARU) acoustic display modes and their associated annotation
- Displays alerts

- Displays and processes responses from cues
- Displays tableau and processes responses from tableau modification
- Enters and updates acoustic contacts and fixes
- Utilizes track ball inputs

3.2.1.2.8 SRX Frequency Control (DU08)

- Provides sonobuoy receiver initialization
- Regulates sonobuoy receiver input/output
- Assigns sonobuoy receivers to ADP processor channels

3.2.1.2.9 ATR Control (DU09)

- Provides analog tape recorder initialization
- Provides operator interface to analog tape recorder
- Provides analog tape recorder control

3.2.1.2.10 ADP Passive Contact Classification (DU10)

- Places MPD in classify display mode
- Enables operator to change the classify mode RF
- Enables operator to modify stored data for the classify RF
- Provides various related processing, alert, and display functions

3.2.1.2.11 ADP Passive Contact Maintenance (DU11)

- Provides the contact data matrix which permits the operator to participate in contact maintenance
- Provides functions in the display control matrix to permit the operator to monitor contact data

3.2.1.2.12 ADP Active Ping Control (DU12)

- Controls selection of command active and directional command active sonobuoy systems
- Controls sequence of pinging

TABLE 3-4. DECOMPOSITIONAL UNITS

DECOMPOSITIONAL UNIT (DU)		ASSOCIATED SENSOR/SUBSYSTEM
NOMENCLATURE	NO.	
ESM Intercept Maintenance	01	Electronic Surveillance Measures (ESM)
ESM Frequency Control	02	
ESM Classification	03	
ESM Contact Maintenance	04	
ADP Mode Control	05	Acoustic Data Processing Subsystem (ADP)
ADP Data Management	06	
ADP Display Control	07	
ADP Passive Contact Classification	10	
ADP Passive Contact Maintenance	11	
ADP Active Ping Control	12	
ADP Active Contact Maintenance	13	
SRX Frequency Control	08	Sonobuoy Receiver Subsystem (SRX)
ATR Control	09	Analog Tape Recorder (ATR)
MAD Display Control	14	Magnetic Anomaly Detection Subsystem (MAD)
MAD Data Management	15	
MAD Signal Feature Recognition	16	
MAD Contact Maintenance	17	
RADAR Display Control	18	Radar (RADAR)
RADAR Data Management	19	
RADAR Contact Maintenance	21	
Display Operator Control	22	S-3A Tactical Command and Control
Tactical Display Management	23	
Display Refresh Management	24	
Keyset Control	25	
Contact Tracking	43	
Tactical Coordination	44	
NAV Position Control	26	Navigation Subsystem (NAV)
NAV Position Calculation	27	
NAV Data Management	28	
NAV Track History	29	
Aircraft Steering Control	37	



TABLE 3-4. DECOMPOSITIONAL UNITS (Continued)

DECOMPOSITIONAL UNIT (DU)		ASSOCIATED SENSOR/SUBSYSTEM
NOMENCLATURE	NO.	
COMM Receive	30	Communications (COMM)
COMM Control	31	
COMM Data Management	32	
COMM Transmit	33	
Sonobuoy Selection	34	Search Stores Control Subsystem (SESCON)
Sonobuoy Release	35	
Sonobuoy Drop-Point Calculation	36	
Ballistics	20	Armament Control Subsystem (ARMCON)
Weapon Release	38	
Weapon Drop Point Calculation	39	
Weapon Selection	40	
SRS Control	41	Sonobuoy Reference Subsystem (SRS)
Sonobuoy Position Calculation	42	
FLIR Contact Maintenance	45	Forward-Looking Infrared Subsystem (FLIR)
FLIR Display Control	46	
FLIR Stabilization	47	

3.2.1.2.13 ADP Active Contact Maintenance (DU13)

- Provides for entering and updating active acoustic contacts

3.2.1.2.14 MAD Display Control (DU14)

- Permits the operator to display real-time MAD signal data
- Establishes the controlling operator and availability of MAD functions

3.2.1.2.15 MAD Data Management (DU15)

- Obtains input data from MAD subsystem and stores it in a file

3.2.1.2.16 MAD Signal Feature Recognition (DU16)

- Initiates feature recognition processing
- Performs MAD data analysis

3.2.1.2.17 MAD Contact Maintenance (DU17)

- Permits the operator to enter a MAD fix into the system based on the position of the horizontal cursor on MAD display
- Permits the operator to accept an automatically detected MAD signal as a MAD fix and enter it into the system

3.2.1.2.18 Radar Display Control (DU18)

- Permits the operator to control radar set configuration by selection of appropriate control function
- Provides radar display and control
- Provides scan converter control
- Provides antenna control

3.2.1.2.19 Radar Data Management (DU19)

- Provides all data communication to and from the radar subsystem

3.2.1.2.20 Ballistics (DU20)

- Determines speed and motion of buoys and weapons when in flight

3.2.1.2.21 Radar Contact Maintenance (DU21)

- Permits the operator to enter radar fixes into the system

3.2.1.2.22 Display Operator Control (DU22)

- Provides operator aids, functions, and requests to enable tableaux, cues, and alerts to be displayed

3.2.1.2.23 Tactical Display Management (DU23)

- Manages the tactical plot area of each display
- Provides manual entry functions to assist the operator in processing inputs
- Provides the operator with various control functions, including tableau and symbology control

3.2.1.2.24 Display Refresh Management (DU24)

- Formats the data buffer and transfers data to all tactical displays maintaining an updated presentation

3.2.1.2.25 Keyset Control (DU25)

- Interprets key depressions of operator's Integrated Control Subsystem (NCOS) trays
- Sends lighting commands to NCOS trays and table
- Schedules associated units
- Processes trackball inputs from NCOS trays

3.2.1.2.26 NAV Position Control (DU26)

- Initializes navigation system
- Activates dead-reckoning processing
- Activates and modifies Zulu timekeeping
- Enters and displays navigation symbols and grid coordinates
- Initiates and terminates the tactical mode of navigation
- Allows operator correction of aircraft tactical position
- Modification and computation of bias velocity
- Allows operator correction of tactical positions of buoys, contacts, and fixes
- Sets position of buoy and contacts to last aircraft position
- Corrects aircraft geographic position
- Operator inputs of magnetic variation or Carrier Aircraft Aligned Inertial Navigation System (CAINS) data

3.2.1.2.27 NAV Position Calculation (DU27)

- Computes aircraft position from NAV sensor data
- Stores latitude/longitude for output to the inertial navigation subsystem (INS) and the altitude and heading reference subsystem (AHRS)

3.2.1.2.28 NAV Data Management (DU28)

- Receives, converts, maintains, and stores signal data from the NAV subsystem

3.2.1.2.29 NAV Track History (DU29)

- Allows the operator to display the ASW support aircraft carrier (CVS) computed position
- Maintains CVS position, course, and velocity
- Maintains the track history of aircraft

3.2.1.2.30 COMM Receive (DU30)

- Receives, converts, and maintains data from COMM subsystem

3.2.1.2.31 COMM Control (DU31)

- Initializes COMPAC module
- Transmits all data on link in one transmission during data silence
- Retransmits the data terminal set (DTS) and deactivates Net
- Configures hardware before starting Net
- Configures Net in receive-only silence mode
- Activates Net and performs synchronize and control functions
- Monitors status of COMM subsystem for faults and configuration changes

3.2.1.2.32 COMM Data Management (DU32)

- Allows the operator to assign data to net, assign special points of interest, recenter display, specify engagement status, assign contact status to buoy, and assign the identification friend or foe (IFF) selective identification feature (SIF) code to track on net
- Transfers COMM data base to another aircraft
- Transmits emergency alerts of data on Net
- Corrects and updates aircraft position
- Inhibits remote data from display
- Displays link track numbers
- Unlinks data from net, destroys COMM data, and purges old data



- Formats on-station message
- Acknowledges order received and answers it
- Puts emergency message on Net

3.2.1.2.33 COMM Transit (DU33)

- Maintains, converts, and transmits communication data to the COMM transmitters

3.2.1.2.34 Sonobuoy Selection (DU34)

- Resets and initializes SESCON module
- Allows the operator to modify parameters in the search stores inventory
- Allows the operator to select one of the following buoys for release: low-frequency acquisition and ranging (LOFAR), directional LOFAR (DIFAR), command activated sonobuoy system (CASS), directional CASS (DICASS), bathythermograph (BT), range only (RO), sound under-water signal (SUS) or Smoke
- Allows the operator to enter a pattern of fly-to points into the system and to select a sonobuoy for release at each point

3.2.1.2.35 Sonobuoy Release (DU35)

- Performs release of sonobuoy from aircraft and insertion of buoy into system, executing upon demand
- Updates search stores inventory

3.2.1.2.36 Sonobuoy Drop-Point Calculation (DU36)

- Calculates the fly-to point of a sonobuoy

3.2.1.2.37 Aircraft Steering Control (DU37)

- Allows the operator to assign a constant radius aircraft orbit about a chosen point
- Initiates circular orbit about an active range buoy with an orbit radius equal to and varying with target range
- Computes and displays the intercept point between target and aircraft
- Allows the operator to enter and display fly-to points

- Computes synthetic fly-to points for aircraft control in special steering patterns
- Determines desired ground track in directing aircraft to fly-to point and furnishes data to the navigation data repeater and converter (NDRC)

3.2.1.2.38 Weapon Release (DU38)

- Releases selected weapon(s) on demand

3.2.1.2.39 Weapon Drop Point Calculation (DU39)

- Calculates the fly-to point of a weapon

3.2.1.2.40 Weapon Selection (DU40)

- Initializes the ARMCOS function
- Selects bomb, cluster bomb, mine, practice bomb, special weapon, or torpedo for release

3.2.1.2.41 Sonobuoy Reference Subsystem (SRS) Control (DU41)

- Selects/deselects sonobuoys for SRS processing
- Initiates SRS buoy positioning processing
- Monitors SRS for hardware failures
- Computes calibration factors for SRS processing

3.2.1.2.42 Sonobuoy Position Calculation (DU42)

- Computes sonobuoy positions
- Computes sin/cos of SRS on-top position indicator (OTPI) bearing
- Computes calibration factors for angle-measuring equipment (AME) and direction-measuring equipment (DME) data
- Does preliminary processing and scheduling and requests I/O for SRS calibration and normal processing routines

3.2.1.2.43 Contact Tracking (DU43)

- Updates intercept, predicts position and active sonobuoy-to-MAD attack tactics (ASMAT) contacts
- Updates probability contours
- Updates tracks

3.2.1.2.44 Tactical Coordination (DU44)

- Enters visually identified, operator specified, and computed fixes (the latter at intersections of bearing lines or range circles associated with two contacts)
- Displays probability ellipse about a fix, track, or in-flight training program (IFTP)
- Displays circle about target position containing possible locations of target when aircraft arrives at position
- Enters local data into system
- Constructs target track based on fixes
- Displays fix positions for a generated track
- Allows operator to modify track number and category identification of fix or track
- Estimates current position of target and computes and displays predicted position for a given target at specified time
- Calculates parameters for ACTIVE RANGE tableau
- Computes and displays a suggested pattern for CASS and RO buoy deployment
- Allows operator to acknowledge entry of new contacts

3.2.1.2.45 FLIR Contact Management (DU45)

- Enters and maintains FLIR fix

3.2.1.2.46 FLIR Display Control (DU46)

- Initializes/disables computer control of FLIR hardware
- Activates FLIR cooling fans and sensors
- Extends/retracts FLIR outside/inside aircraft
- Displays FLIR mode on MPD
- Adjusts brightness/contrast and modifies image resolution
- Reverses FLIR polarity

3.2.1.2.47 FLIR Stabilization (DU47)

- Pans a given sector in manual track mode
- Initiates the FLIR auto track mode
- Performs FLIR output and positioning commands

### 3.2.2 Control Requirements

Decompositional elements have now been defined for the BASIC application. The next step in the methodology is the synthesis of a control structure which relates the DU's to one another. The objective of this paragraph is to define such a control structure.

The first step is to define a comprehensive set of functions necessary for the control of data processing software and hardware in an avionic system with operational requirements similar to those of the S-3A. These functions will provide a baseline set of control requirements suitable for incorporation into the BASIC concept.

#### 3.2.2.2 Characteristics of Avionic Computer Processes

Before developing a set of control functions, the nature of the controlled processes must be well understood. One primary characteristic of avionic processes is that they are predefined and do not change during a particular mission. Only the sequence in which the processes execute is not known beforehand. Thus, much of the environment and interfaces required by a particular process can be defined at system-generation time; this relieves the run-time control mechanism of these burdens. The other primary characteristic of avionic processes is that they are all interrelated to some degree and that they all contribute to the execution of a single, overall function, called the mission. For this reason, the efficiency of the control structure must be measured in terms of contribution to mission success, which is a complex and ill-defined measure, rather than in terms of percentage of overall time required for control, which is a standard measure for a batch processing environment.

In general, avionic processes execute in one of three modes: synchronous, demand, and background. In the synchronous mode, processes are dispatched periodically and are expected to be complete before the next dispatch. Most of the processing time in avionic systems is used by synchronous processes. In the



demand mode, processes are dispatched as a result of an interrupt, usually generated by a keyset depression. Most of the tasks executed by an avionic system are demand mode tasks. They do not, however, constitute a heavy load on the processing resources. Background tasks are low-priority tasks executed whenever the processor is not otherwise occupied. Avionic processes usually require more program storage than data storage, and the data is usually updated periodically. The ratio of program storage required to data storage required runs from 85 percent for F-15 fighter aircraft to 50 percent for E-2C surveillance aircraft.

3.2.2.2.1 Synchronous Processes. Synchronous processes are scheduled and dispatched periodically by the control mechanism. In general, they operate on sensor data which is always being updated at a particular rate determined by the sensor. It is therefore necessary that the periodic processes utilize the data before it gets overwritten by new data.

In the avionic environment, time is measured in terms of cycles. Each cycle is a small increment (50 msec for the S-3A), and all time is measured in integer numbers of cycles. Thus, a periodic process is dispatched every N cycles and must be complete within a certain number of cycles. This method allows the control mechanism two options if a process cannot be dispatched or completed on time; either the start of a new cycle can be delayed, or the process can be aborted.

In the S-3A, the synchronous tasks account for almost all the processor loading, while the actual number of synchronous tasks is only 10 percent of the total number of tasks executed by the system.

3.2.2.2.2 Demand Tasks. Demand tasks are most often executed in response to an operator-generated interrupt and generally result in a change to the display associated with that operator. Demand tasks cannot be prescheduled, as can synchronous tasks. This presents a more difficult problem to the control

mechanism, since it must recognize the interrupt, decide which process is to be executed, interrupt and save the state of the currently executing process, dispatch the demand task, and restore the previous process when the demand task is complete.

On the average, demand tasks require about 2 msec of processing time out of every second. However, out of the 331 tasks identified for the S-3A processing system, 295 (89 percent) are demand tasks, while only 36 are synchronous.

The performance measure of the control structure with respect to demand tasks is the time which is required to start execution of the process after the interrupt is received. This time can be quite critical, especially if the demand interrupt is caused, not by an operator action, but by the identification of certain parameters in sensor data. These data may require additional special processing before they are overwritten; thus, speedy dispatch of the appropriate process is necessary.

3.2.2.2.3 Background Tasks. Background tasks are executed when no demand tasks have been requested and no synchronous task is ready for execution. Test programs, bus polling processes, and certain functions of the executive are normally background processes.

3.2.2.2.4 Interrupt-Activated Processes. Occasionally, an avionic process operates in a combination of demand and synchronous modes. This occurs when an interrupt causes a synchronous process to become active. After activation, the process behaves exactly as a synchronous task until another interrupt deactivates it, after which it is no longer scheduled on a periodic basis. These interrupt-activated processes are usually associated with the operator's desire to see a particular display format for a time and then return to his normal display format. Thus, the process is synchronously executing only as long as the operator desires to display the results. Otherwise, the process is not executing.

3.2.2.2.5 Functions. Avionic processes, in a very general sense, are concerned with the display of data gathered by the sensors. A typical sequence of processes for a mission sensor would input sensor data; filter it to remove noise; compensate for the position, heading, motion, and physical geometry of the aircraft; decide if a detection has occurred; and, if it has, format the data for display on a cathode ray tube (CRT). These processes usually perform numerous trigonometric and averaging calculations. Data accuracy and resolution is usually in the 8 to 16 bit range. Considerable man-machine cooperation is necessary to detect and classify objects perceived by the various aircraft sensors, thus leading to the large number of demand tasks required. Navigation and communication processes also filter data received from sensors, compensate for aircraft parameters, and display the results. These processes do not, however, require detection and classification functions.

### 3.2.2.3 Avionic Processing Environment

There are several important characteristics of the processing environment that must be considered in the definition of a control structure. A process executes in an environment which can partly be controlled by the system designer through definition of the system architecture and is partly beyond the designers' control because of weight, cost, and state-of-the-art constraints. For example, the designer would normally not choose to increase radar reliability by implementing dual surveillance radar systems, such as that on the E-2C, since the weight and cost penalty would be too great. Similarly, no amount of architectural manipulation will enable an avionic system to detect and classify targets beyond the range of any of its sensors. The primary impact of these weight, cost, and state-of-the-art constraints on the control mechanism is in the areas of signal processing and fault tolerance.

Although there is no theoretical difference between signal and data processing, there is an undeniable practical and architectural difference. Signal processing must be performed at a speed much greater than can be attained by a general-purpose data processor. As a result, signal processing devices are

generally high-technology devices whose internal architecture is matched with the sensor(s) it services. Signal processors are larger and more expensive than data processors and require more power and a greater attention to physical geometry because of their higher speeds for internal data transfer. They are architecturally geared for limited applications. As such, the processing system designer has very little freedom in the number and type of signal processors he may use. For this reason, signal processors have been removed from consideration as part of the data processing system and will henceforth be considered as part of the particular sensor subsystem in which they are used.

The ability of a system to recover from a fault is also constrained by weight and cost factors. As stated before, it is unrealistic to duplicate large and expensive sensors to provide better fault tolerance; therefore, it is unrealistic to expect to maintain full performance in the face of any fault that may occur. Likewise, it is inefficient to provide a computer with fault tolerance capability such that the mean time between failures (MTBF) of the computer exceeds the reliability of any of the sensors it may service by more than a factor of five or six.

In cases like this, it is required that the system be able to lose the capability provided by the sensor without a complete system crash. That is, the system must be capable of providing the maximum performance possible without the data normally provided by the failed sensor. Such graceful degradation can be provided for in the design of the architecture.

Many aspects of the processing environment, however, are under the control of the processing system designer. Complete recovery from a processor fault can be attained through the proper use of redundant processing hardware and interface paths. Flexibility and growth can be built into the system so that increased or modified requirements can be met at minimum costs, or new technology can be easily incorporated to improve performance.



With this understanding of the avionic processes and their environment, a set of control functions for the architecture may now be defined. The following paragraphs describe the control functions recommended for the BASIC processing system.

#### 3.2.2.4 Functional Requirements for Control of a Multiple Computer System

The functions necessary for control of a processing system can conveniently be broken into two categories; local control, which deals with the internal actions of each computer; and global control, which coordinates and manages the interactions among the various computers. This division and the functions required for each category are shown graphically in Figure 3-3 and are described in detail below.

##### 3.2.2.4.1 Local Control Functions

- a. Initialization. The computer should be capable of putting itself into a state in which it can perform self-check and inform an external source of the results of the self-check. If the self-check indicates a working condition, then the computer should be capable of performing any of the following subfunctions
  - (1) Reload — This function performs a master clear of all necessary computer status and register contents, and executes a resident bootstrap load program.
  - (2) Cold Start — After self-check, the computer executes a master clear and initializes any necessary control information from data loaded via the bootstrap loader. A prespecified process then begins execution.
  - (3) Autostart — This function reinitializes the computer after an interrupt has occurred because of a hardware failure (for example, low power). Data for reinitialization shall have been stored in non-volatile memory by the interrupt service routine.

- b. Timekeeping

A timekeeping mechanism is necessary for this internal use of application processes and for the scheduling of synchronous programs. In addition, system time in terms of cycles must be kept to aid in coordination of interdependent processes executed on different computers.

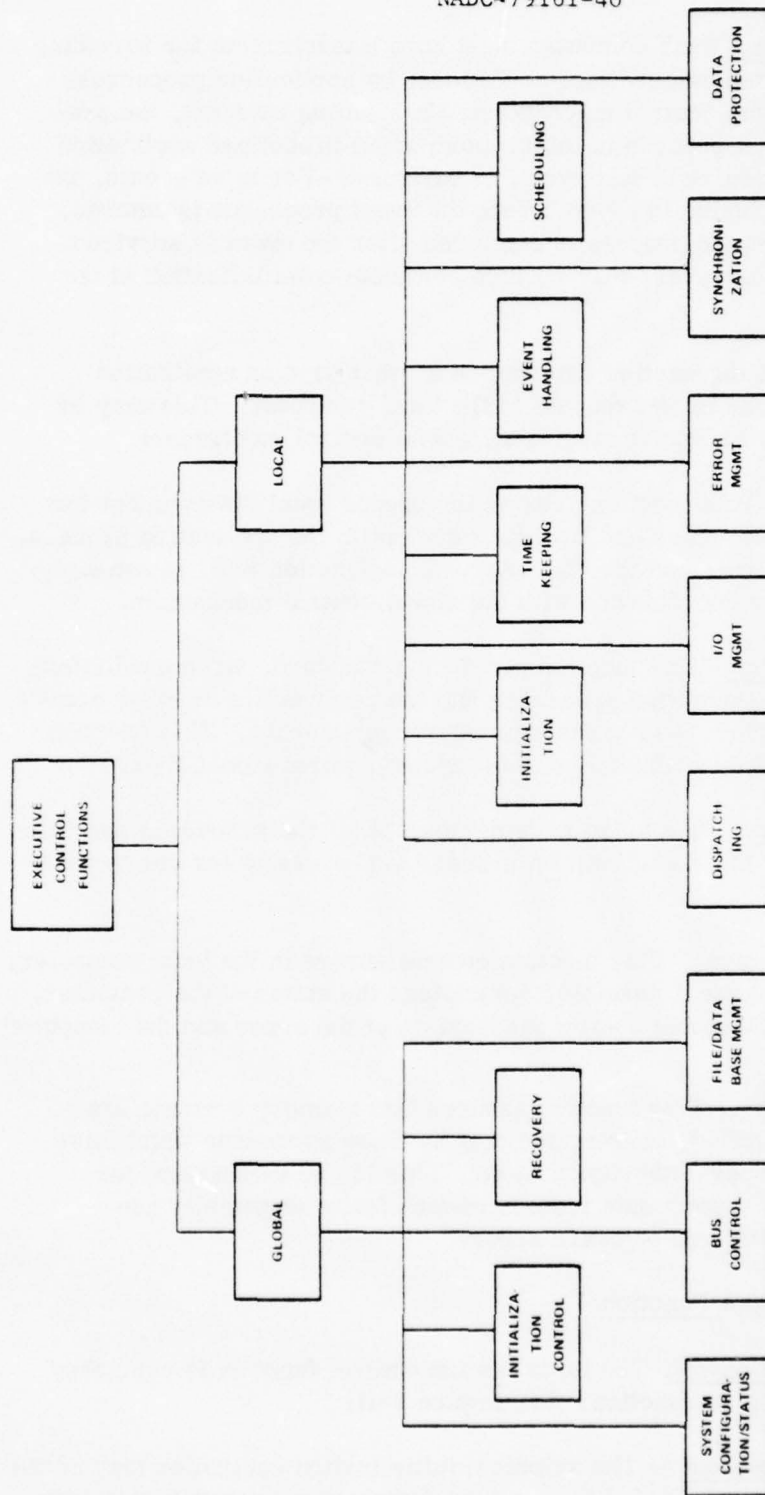


Figure 3-3. Executive Control Functions

- c. Event Handling. Each computer must have a mechanism for servicing events generated by hardware conditions, by application processes, and by the global control mechanism. In handling an event, the presently executing process is interrupted; and a predefined application process, particular to that event, is executed. For most events, the state of the machine is saved before the event processor is entered, and the interrupted process is continued after the event is serviced. Some events, however, may require complete reinitialization of the computer.
- d. Scheduling. This function determines at what time an application process is ready to be executed in the local computer. This may or may not involve interacting with the global control mechanism.
- e. Dispatching. This function creates the proper local environment for an application process and transfers control to the application process, which then executes on the computer. This function will, in some cases, require coordination with the global control mechanism.
- f. I/O Management. This function provides a standard, structured interface between application processes and the peripherals or other computers with which these processes must communicate. This function must be closely coordinated with the global control mechanism.
- g. Synchronization. This function determines if all the resources and data necessary for the execution of a process are available for use by that process.
- h. Error Management. This function detects errors in the local computer, corrects the error if possible, determines the status of the computer, and informs the global control mechanism of the error and the computer status.
- i. Data Protection. This function assures that memory contents are accessed, modified, or executed only by those processes which have been given proper authority to do so. This is one mechanism for protecting the system data from hardware faults (especially non-recurring faults) and software errors.

#### 3.2.2.4.2 Global Control Functions

- a. Initialization Control. The initialization control function is composed of the following subfunctions (See Figure 3-4):
  - (1) Initial Checkout — The avionics' initialization controller (any of the processors) is loaded from a load device containing unit test software designed to verify the integrity of the system. Self-test is

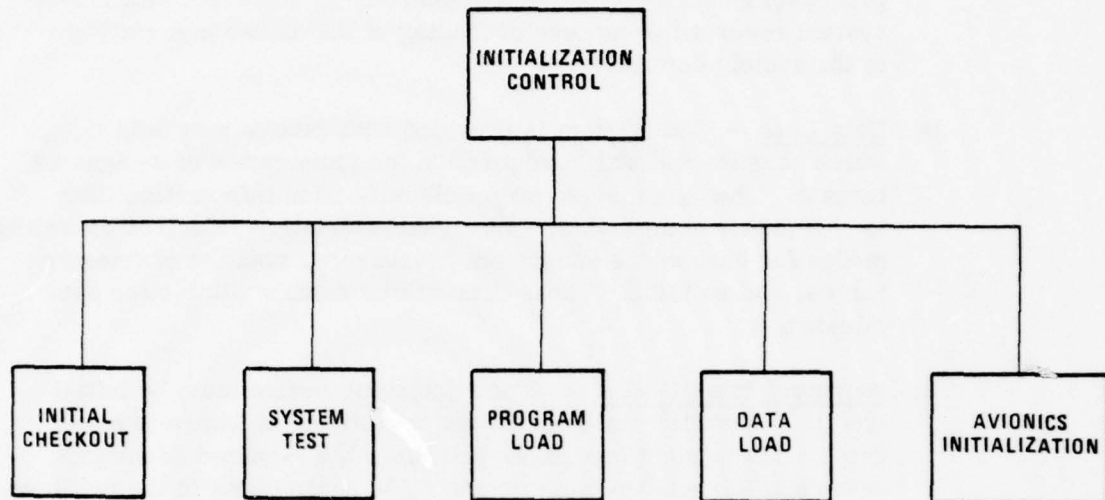


Figure 3-4. Initialization Control Function Hierarchy

performed internally by the controller and includes the CPU and external memories. Loop tests are conducted between the controller and the individual avionic subsystems in order to verify the communication links. Configuration data, which establish the various operating modes of the subsystem elements, are loaded from the initialization controller into each of the avionic subsystems. The initial checkout process includes a verification that a fault is sensed and reported to an appropriate display. The system controller directs each subsystem processor to execute its built-in test function and report the results back for appropriate display. The system controller tabulates the results received from the initial checkout process and reports the checkout as being either Go or No-Go. A No-Go report will effect a system shutdown.

- (2) System Test — The system test subfunction requires a hardware diagnostic program to be run for the total system, including subsystem interfaces and each subsystem. The system test verifies the proper operation of the equipment in the system and reports as to the proper operation or malfunctioning of a specific subsystem. The system test isolates the problem within the malfunctioning subsystem to a specific subunit or group of pages that can be replaced in order to rectify the problem.



- (3) Program Load — Following a successful initial checkout and system test, the operational software is loaded into each of the subsystem processors with their respective operational software. Each subsystem reports the successful loading of its operational software to the system controller.
  - (4) Data Load — The system is provided with data on magnetic tape, which permits optional configuration for prosecution of designated targets. The data include target classification information (that is, frequency components, radar characteristics), desired operating modes for each of the subsystem processors, position of friendly forces, and so forth. These data will be made available for each mission.
  - (5) Avionics' Initialization — The avionic subsystem must be initialized to a specified state before the operational software is executed. The system controller will issue the required commands to each of the subsystems to place these subsystems in the required state. Failure to perform this initialization function will cause the software operating system to report a software system error.
- b. Recovery (Figure 3-5) The recovery function provides for the following contingency: when a subsystem fails and this failure is recognized and isolated by the in-flight performance monitoring software, the system can be reconfigured either automatically or manually into a degraded mode; and the mission can proceed. The system will select the desired configuration and proceed with the initialization function. At the completion of the initialization process, the system will enter the information contained in the checkpoint status data base. This data base contains subsystem configuration status information, which includes the operating mode for each processor and stores inventory, and problem data, which include tactical display inputs, sensor classification coefficients, and navigation computations. At the completion of the recovery process, the system will display a message that the recovery has been successful along with the resultant degraded configuration.
- c. System Configuration (Figure 3-6). The system is required to maintain status with each of the subsystem processors to verify that the communication link is operating properly and that the subsystem is properly configured. The system will format and receive messages, as scheduled by the executive, to verify the configuration status of each of the subsystems and peripherals. The system also has the capability to reconfigure any of the subsystems to a new configuration.

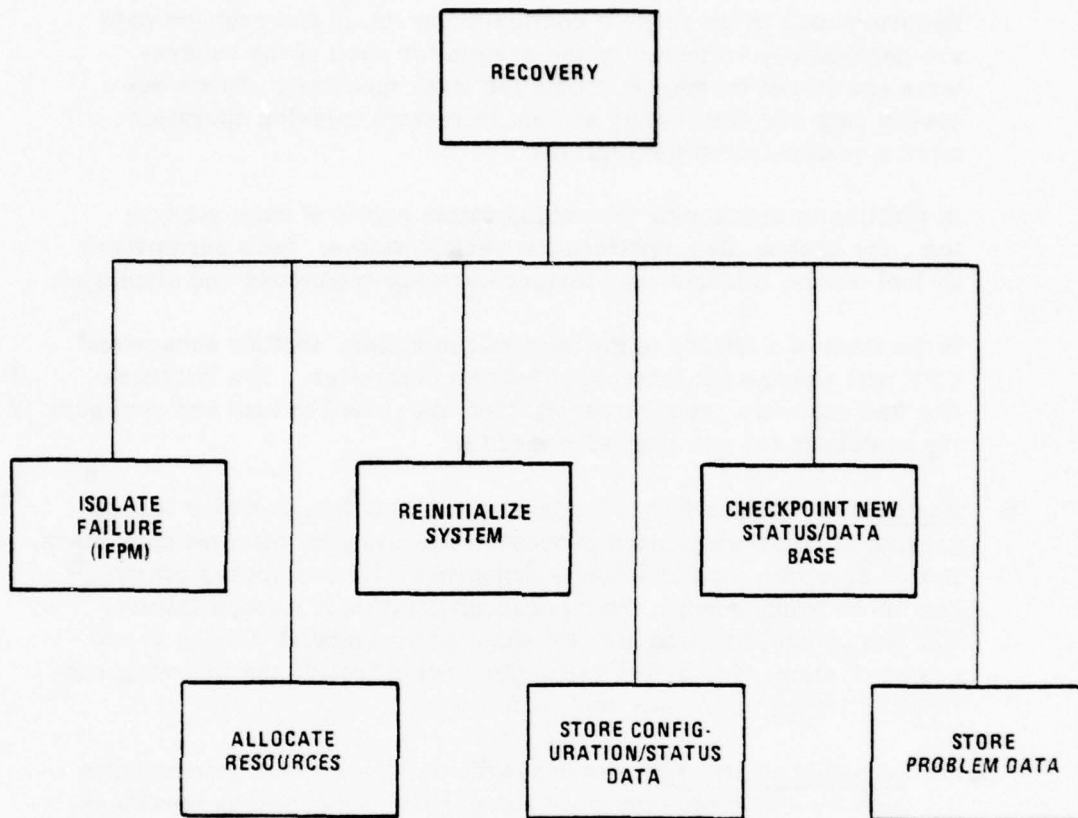


Figure 3-5. Recovery Control Function Hierarchy

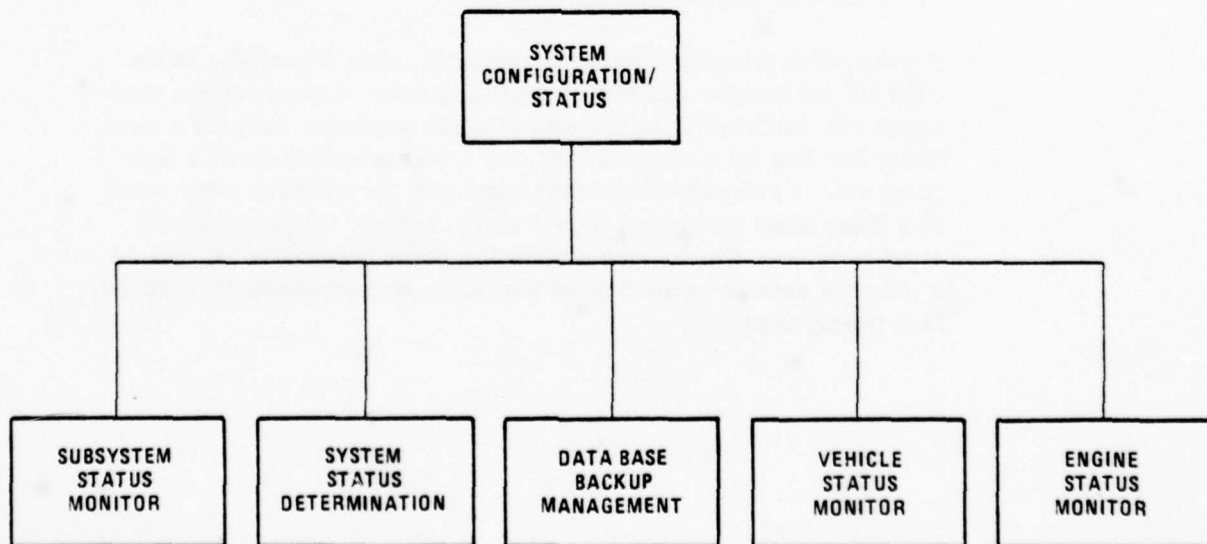


Figure 3-6. System Configuration/Status Control Function Hierarchy

Recovery data in the form of configuration status and problem data are periodically collected by the system for each of the subsystems and stored on magnetic tape (or other medium). These recovery data are used by the system to restore mission operation after a system/subsystem failure.

In addition to monitoring the configuration status of each subsystem, the system also monitors the vehicle status. Such parameters as fuel status, attitude, and temperature are monitored and displayed.

In the case of a failure of the system controller, another subsystem CPU will assume the function of system controller. The initialization and recovery procedures will then take place to load and configure the system in the new degraded mode.

- d. Bus Control (Figure 3-7). The bus control function provides for the passing of messages among processes executing in different computers. In this capacity, this function is concerned with bus access control, bus error management, and bus reconfiguration in case of failure. The bus control function also provides each computer access to necessary system-wide global variables. Specifically, the following subfunctions make up the bus control function:

- (1) Message Processing — Messages passed between avionic units may be classified into three categories: synchronous messages, asynchronous messages, and critically timed messages. It is the responsibility of the bus control function to assure that these messages are transmitted and received correctly within the time limitations of their particular category.

Synchronous messages must be transmitted on a periodic basis (that is, an integer number of cycles apart). Asynchronous messages are initiated upon request of some process, such as a request for data by a demand task, or a status update from a text program. Critically timed messages are those which must occur at a finer time resolution than a single cycle. The bus control must provide a timing mechanism for these messages as well as a priority access to the bus so that their transmission is assured in a timely manner.

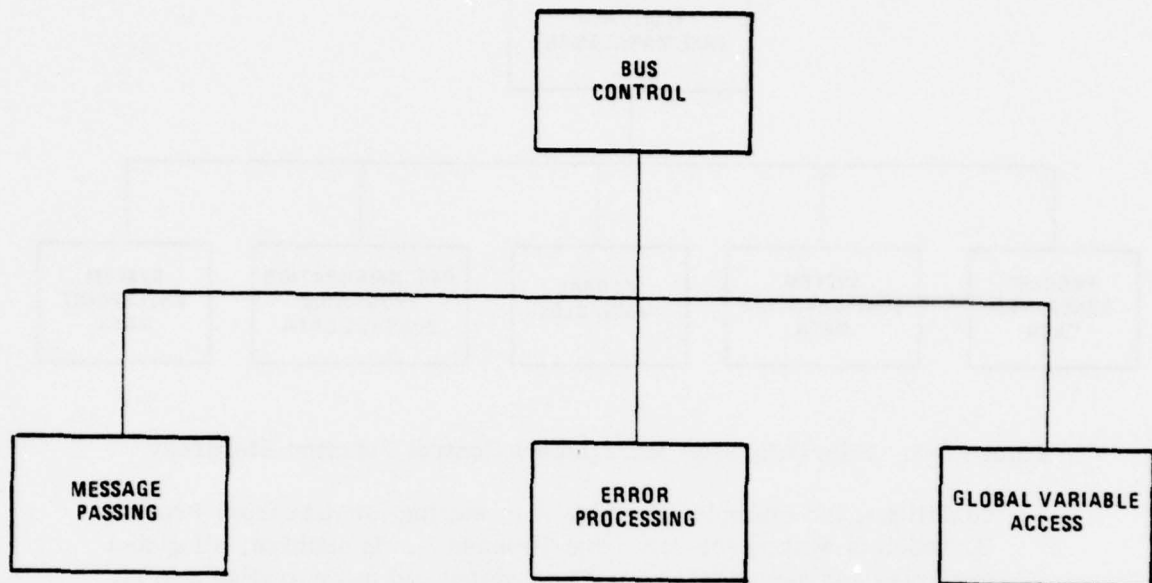


Figure 3-7. Bus Control Function Hierarchy

- (2) Error Processing — This subfunction is required to detect, isolate, and recover from faults in the message or message path. The bus control must be able to identify faulty messages and either correct them or take some action to assure that they are not taken as correct by the rest of the system. Recovery may be by use of error correction mechanisms, retry, or cancellation of the message and reconfiguration using available excess hardware and software resources. The bus control function must be capable of detecting, isolating, and recovering from faults within its own mechanism.
- (3) System Variable Access — The bus control function must provide each unit on the bus with the capability to access any global variables and semaphores. This is considered a different category from simple message passing since, depending on implementation techniques, the location of these variables and semaphores may not be known previous to run time.
- e. File/Data Base Management (Figure 3-8). The system must have a mechanism by which data generated by one process in one computer can be made available to another process in another computer when needed. Appropriate protection mechanisms must be implemented to assure proper updating of data and the avoidance of "deadly embrace"



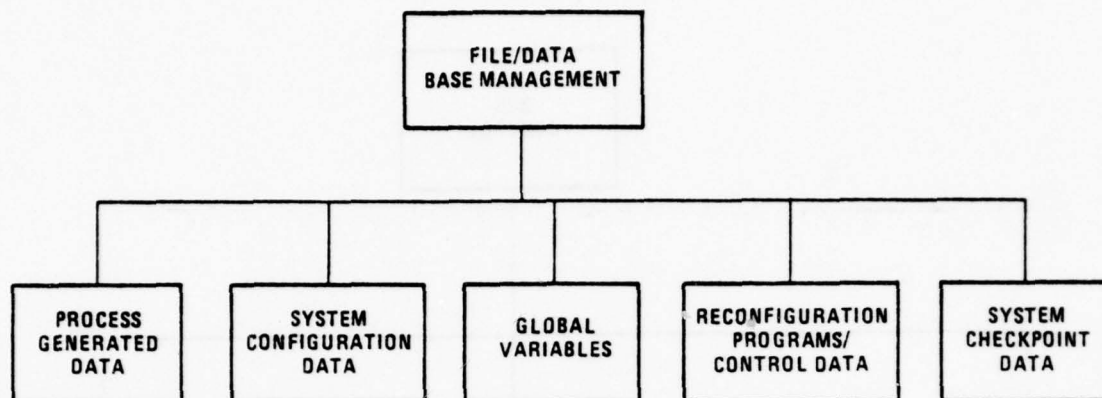


Figure 3-8. File/Data Base Management Control Function Hierarchy

conditions; for example, Process A is waiting for data from Process B, which is waiting for data from Process A. In addition, all global variables and semaphores must be updated and made available to all processes. Reconfiguration programs and control data must be available for use if required. Finally, system checkpoint data, status, and configuration must be maintained.

- f. Synchronization. As in the local control mechanism, this function resolves conflicts that may arise when two or more processes require the use of a single resource. Semaphores are the usual mechanisms used by the synchronization function.
- g. Dispatching. This function, again as in the local control mechanism, determines when the global conditions necessary for the execution of a particular process have been satisfied.
- h. Event Coordination. If an application process causes an event, that event is normally serviced by another process. It is the responsibility of the event coordination function to see that this second process is queued for dispatch (regardless of where it is in the system) and to pass any necessary messages between the event-causing process and the event-servicing process.

## SECTION 4

## APPLICATION STRUCTURE

A set of application DU's and the functions for controlling their execution have been defined. Now implementation techniques must be developed such that these functions can execute and perform the mission optimally with respect to a set of predefined assessment criteria. These assessment criteria are chosen according to the demands of the application.

The following paragraphs define a very high-level, generic structure that will act as a baseline for the architecture. Assessment criteria for the BASIC application will be defined and justified, and implementation techniques for the baseline structure will be recommended, such that the generic structure will become better defined and applicable to the BASIC mission.

## 4.1 BASELINE GENERIC STRUCTURE

An avionic system can generally be considered as an application function comprising two major elements. These elements include a complement of sensors/subsystems, which perform the search, detection, position determination, and communication functions of a mission, and a processing system fulfilling the corresponding information processing functions. Figure 4-1 characterizes a generic application function architecture in terms of sensor/subsystem elements, processing system elements, and an interconnection structure.

T.O. Wolff's paper, "Improvements In Real Time Distributed Control" (contained in Reference 9), indicated that there are only four basic strategies at the disposal of system architects for attaining higher performance.

- a. Redefinition of the problem in more efficient form — algorithm development
- b. Development of functionally specialized machines
- c. Improvement in raw speed of components
- d. Application of system and machine architectures which support concurrency

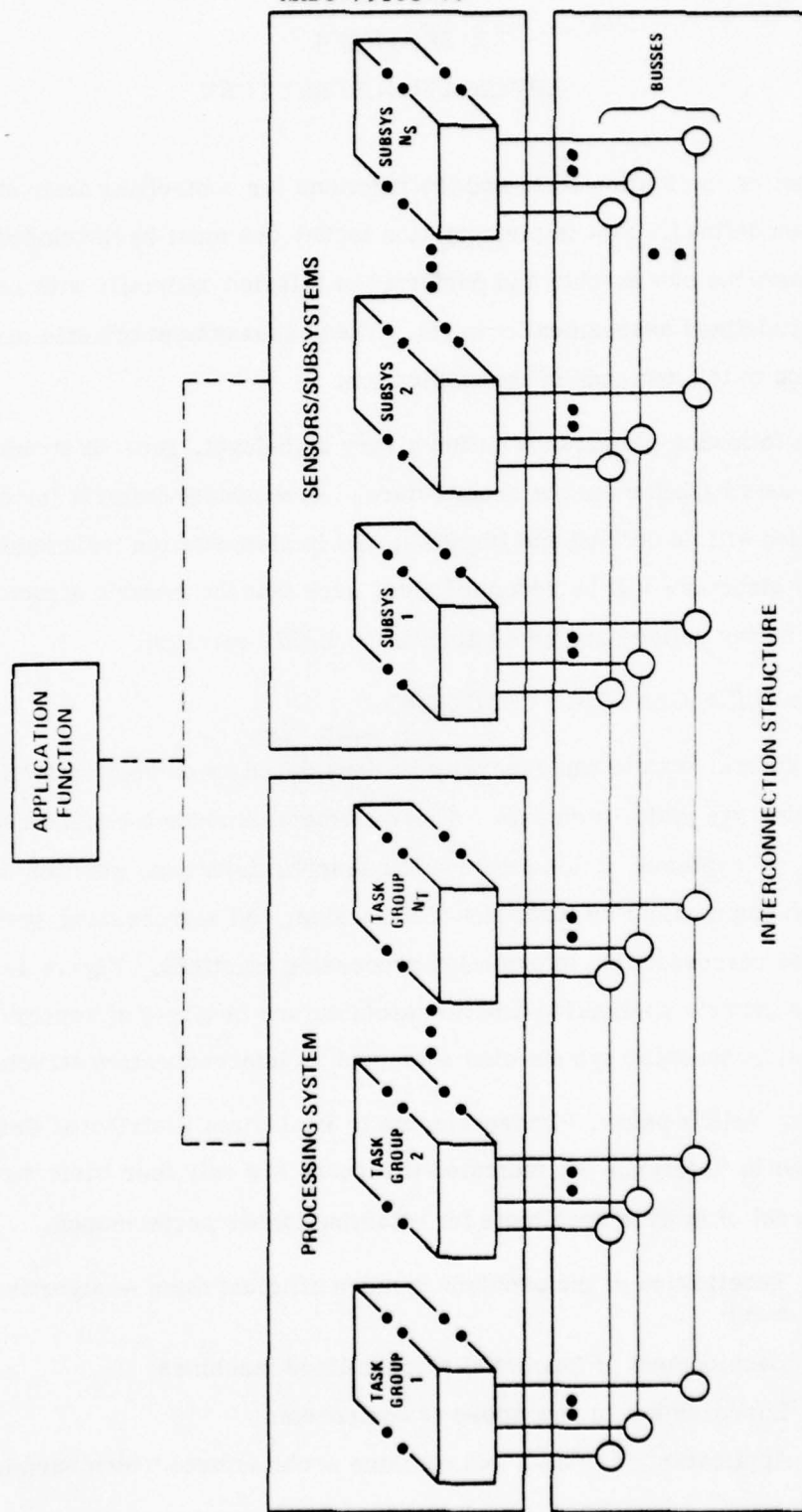


Figure 4-1. Generic Application Function Architecture

Since the first two strategies depend greatly on the application, the gain is moderate, concentrating on only a part of the overall system. The third approach is limited by the state of the art in solid-state devices at any point in time. As a result, the immediate goal of most system architecture work is the fourth strategy, that is, to find methods of multiplying performance by creating systems which exploit parallel, concurrent, or simultaneous execution of tasks in multiple processing elements.

The architecture depicted in Figure 4-1 is organized to employ multiple processing elements by partitioning the processing system function into related task groups. The interconnection structure provides a mechanism which permits any element to communicate with any other. As implied in Figure 4-1, growth in the sensor/subsystem area to accommodate technological advances or responses to new threat conditions should only have impact on the processing system by either adding subtasks within task groups or adding new task groups and making appropriate adjustments to the interconnection structure.

#### 4.2 ASSESSMENT CRITERIA

Architectures that employ multiple processor organizations are generally classified as distributed systems. Some of the current questions pertinent to the expectations of distributed systems include the following:

- Can the system easily adapt to new functions?
- Does the system provide ease of modular, incremental growth and configuration flexibility for both hardware and software?
- What type of interconnect and control structures is required?
- How does the system detect, categorize, isolate, and recover from failures?

In view of these and other concerns regarding distributed processing architectures, this section attempts to establish general design and assessment criteria guidelines for application in the development of a multiple processor organized avionic information processing system. The extent to which the



potential benefits expected of distributed processing systems can be realized depends on the degree to which the subsequent assessment criteria are achieved.

#### 4.2.1 Flexibility

In the literature, system flexibility is referred to by various terms, such as modularity, expandability, adaptability, and extensibility. Essentially, flexibility can be considered to be the ease with which system function and performance can be changed without altering the basic system design.

##### 4.2.1.1 Design Goals for Flexibility

Although implementing the mechanisms to achieve the desired flexibility implied in Figure 4-1 is subject to various current issues regarding multiple processing systems, the generic architecture shown, nonetheless, serves as a model that is suggestive of design goals for which to strive. An ultimate, overall objective is to derive a single, versatile architecture that has the necessary features for potential application to many platforms. Toward this end, the following architectural features indicate a design philosophy that will facilitate system flexibility:

- a. Technology Adaptation. The architecture should accommodate any technological improvements.
- b. Standardization. System elements should be standardized to the greatest extent practicable for increasing interchangeability and commonality.
- c. Interconnection Structure. The number of hardware resources on any bus should only be limited by the data transmission capacity of the bus. In addition, all processors in the system should be able to communicate with the peripherals of any other processor in the system either directly or through another processor.
- d. Task Intercommunication. Programs from an auxiliary memory should be capable of being transferred over the bus structure as a regular part of an executing operational program without degrading the overall system operation.
- e. Executive Control Structure. The system executive should provide an efficient means for supporting task interaction and permit ease of modifying the applications supported.

#### 4.2.1.2 Flexibility Techniques

As indicated previously, Figure 4-1 depicts the general characteristics necessary for achieving system flexibility. Techniques for implementing these characteristics are, however, subject to many architectural and application considerations. The architectural modularity aspect, for example, must be accompanied by well-conceived interfaces for hardware, software, communications, and control in support of the application function being implemented. Paragraph 4.2.1.1 lists several features that are desirable for enhancing flexibility. The following paragraphs provide a further discussion of these features.

4.2.1.2.1 Technology Adaptation. During the life of a system there will be numerous changes in the nature of tactical operations. Some of these changes may result from changes in the anticipated threat or changes in tactics used to counter existing threats. Certain functions within the system may require updating or replacement by new functions. It follows that the processing system design must be able to accommodate or adapt to such changes in function without hardware redesign or disruption of other ongoing functions in the system. This implies that the system must be able to accommodate software revision and the addition of hardware elements without any changes in the basic system design.

In reference to Figure 4-1, the priority of approaches suggested to accommodate functional changes is, first, to add or modify subtasks within a task group and then to add new task groups. The rationale for initially evaluating the addition or modification of subtasks within a task group is to minimize additional hardware and, consequently, interconnection modifications. However, given a flexible interconnection structure, the addition of processor, memory, or I/O channels should also be accomplished with relative ease.

4.2.1.2.2 Standardization. Standardization is a desirable feature since it implies a homogeneous rather than a heterogeneous architecture and is conducive to simplification. The prime advantages of standardization are interchangeability and

commonality, since the numbers of different types of elements in the system are minimized. Of course, the degree to which standardization can be incorporated is subject to various trade-offs.

One of the trade-off decisions with respect to flexibility is where to concentrate the expandability potential of processing power. Since we are dealing with a multiple processing approach, expandability could be made available by utilizing only partial capabilities of more powerful processors, using the reserve capabilities when adding more tasks; or by employing less powerful processors tailored to particular tasks with little to no reserve capabilities and adding processors for additional tasks when required. However, for complete flexibility, provisions should be made for accommodating any combination of the above to the greatest possible extent.

If the design is confined to the constraints of current Navy standardization programs, analysis of the near-term objectives should consider architectures incorporating elements such as the AN/AYK-14 computer and the MIL-STD-1553A bus, with associated executives. For the long term, however, the architecture should be prepared for adapting to new miniaturized technologies, such as 16- and 32-bit microcomputers as well as fiber optic buses.

4.2.1.2.3 Interconnection Structure. An interconnection structure is concerned with the bus configuration and conventions by which the various system elements are able to communicate. The set of rules by which the various elements of the system communicate is governed by the protocol. The protocol also defines status information to be exchanged and maintains coordination between various asynchronously operating functions. For multiple processing systems, the manner in which the various elements are connected is currently referred to as coupling, of which there are two types. Loose coupling is generally characterized by disjoint primary address spaces, use of bit serial buses, and comparatively smaller interprocess communication bandwidth, where communication among distributed elements is accomplished by message passing. Tight coupling

#### 4.2.1.2 Flexibility Techniques

As indicated previously, Figure 4-1 depicts the general characteristics necessary for achieving system flexibility. Techniques for implementing these characteristics are, however, subject to many architectural and application considerations. The architectural modularity aspect, for example, must be accompanied by well-conceived interfaces for hardware, software, communications, and control in support of the application function being implemented. Paragraph 4.2.1.1 lists several features that are desirable for enhancing flexibility. The following paragraphs provide a further discussion of these features.

4.2.1.2.1 Technology Adaptation. During the life of a system there will be numerous changes in the nature of tactical operations. Some of these changes may result from changes in the anticipated threat or changes in tactics used to counter existing threats. Certain functions within the system may require updating or replacement by new functions. It follows that the processing system design must be able to accommodate or adapt to such changes in function without hardware redesign or disruption of other ongoing functions in the system. This implies that the system must be able to accommodate software revision and the addition of hardware elements without any changes in the basic system design.

In reference to Figure 4-1, the priority of approaches suggested to accommodate functional changes is, first, to add or modify subtasks within a task group and then to add new task groups. The rationale for initially evaluating the addition or modification of subtasks within a task group is to minimize additional hardware and, consequently, interconnection modifications. However, given a flexible interconnection structure, the addition of processor, memory, or I/O channels should also be accomplished with relative ease.

4.2.1.2.2 Standardization. Standardization is a desirable feature since it implies a homogeneous rather than a heterogeneous architecture and is conducive to simplification. The prime advantages of standardization are interchangeability and



commonality, since the numbers of different types of elements in the system are minimized. Of course, the degree to which standardization can be incorporated is subject to various trade-offs.

One of the trade-off decisions with respect to flexibility is where to concentrate the expandability potential of processing power. Since we are dealing with a multiple processing approach, expandability could be made available by utilizing only partial capabilities of more powerful processors, using the reserve capabilities when adding more tasks; or by employing less powerful processors tailored to particular tasks with little to no reserve capabilities and adding processors for additional tasks when required. However, for complete flexibility, provisions should be made for accommodating any combination of the above to the greatest possible extent.

If the design is confined to the constraints of current Navy standardization programs, analysis of the near-term objectives should consider architectures incorporating elements such as the AN/AYK-14 computer and the MIL-STD-1553A bus, with associated executives. For the long term, however, the architecture should be prepared for adapting to new miniaturized technologies, such as 16- and 32-bit microcomputers as well as fiber optic buses.

4.2.1.2.3 Interconnection Structure. An interconnection structure is concerned with the bus configuration and conventions by which the various system elements are able to communicate. The set of rules by which the various elements of the system communicate is governed by the protocol. The protocol also defines status information to be exchanged and maintains coordination between various asynchronously operating functions. For multiple processing systems, the manner in which the various elements are connected is currently referred to as coupling, of which there are two types. Loose coupling is generally characterized by disjoint primary address spaces, use of bit serial buses, and comparatively smaller interprocess communication bandwidth, where communication among distributed elements is accomplished by message passing. Tight coupling

is generally characterized by shared, directly accessible, executable main memory, with comparatively higher primary memory bandwidth and lower inter-processor communication latency.

Referring to Figure 4-1, the circles contained within the interconnection structure represent coupling mechanisms. With respect to flexibility, although all elements are shown and should have the potential for being connected, the actual connection/coupling scheme is determined by information flow and data rates of the application. Consider a loosely coupled configuration in which a bus is assumed to be connected to all elements in the system, and designate it to be a global bus. Provided the bus bandwidth is adequate, the addressing capability of the bus should be able to access all elements, with provisions to add more. Capability should also be provided to distribute the information flow simultaneously among several buses to accommodate increased loads.

Provisions for schemes should also be devised such that communication can be accomplished on a local level with separate buses, so that the global bus load can be relieved and still maintain indirect communication with the global bus to extent necessary. Depending on the data rates, the local bus structures could also be loosely coupled.

The choice between loose and tight coupling is subject to a detailed analysis of the projected application requirements, but implementation of both should be taken into consideration for flexibility purposes. From a standardization point of view, it would be desirable to employ one type of bus structure in an incremental fashion to satisfy application requirements.

4.2.1.2.4 Task Intercommunication. In a multiple processing architecture, tasks will be distributed among various processing elements, as implied in Figure 4-1. Because of the dependency of characteristic functions in tactical applications, the interaction of tasks associated with other processors will be required in addition to those associated with a local processor. Certain common routines that are required by the majority of tasks may not be replicated in all processors but, instead, be centralized in a globally accessible processor.

To facilitate flexibility, the location of resources necessary to execute a task should be transparent to the task. This implies the need for an address translation mechanism which coordinates the use of all resources by task, including the processor and memory to which it is assigned.

4.2.1.2.5 Executive Control Structure. An executive is concerned with a well-conceived control structure that is capable of directing all system resources to interact efficiently in accomplishing the overall application function. The executive is responsible for functions such as the scheduling and dispatching of tasks, resource control, task intercommunications and control, and interrupt handling. In centralized systems, the executive control structure is essentially associated with a single processor. However, multiple processing system architectures require the executive as well as the tasks to be distributed among the various processing elements.

One of the current issues regarding multiple processing architectures is that of load balancing. For purposes of discussion, load balancing can be considered either dynamic or static. Dynamic load balancing is concerned with the dynamic (or automatic) assignment of tasks so that the workload is as equally distributed as possible among the various processors. The control structure necessary to realize this type of architecture is extremely complex and is the subject of many investigative studies. In static load balancing, on the other hand, tasks are preassigned to specific processors at system generation time, so that the location of these tasks is always known. The only time tasks are re-assigned to alternate processors is after failure reconfiguration, and even these assignments are deterministic. The control structure necessary to realize this type of architecture appears to be less stringent than the dynamic, because of the elimination of the dynamic workload distribution mechanism. The static load balancing approach is selected for deriving the architecture in this report because it more nearly matches the nature of the avionic processing problem and because it presents less of a run-time burden on the processing system. Other related issues concern the distribution of system-wide control and the mechanism

by which the operational integrity of the system is assured to be preserved. These and other control structure design aspects are discussed elsewhere in this report.

#### 4.2.2 Fault Tolerance/Recovery

##### 4.2.2.1 Definition of Terms

Fault tolerance is an attribute of an information processing system that enables the system to continue expected behavior after the appearance of certain classes of faults (physical, man-made, or both) that would otherwise force the system into an error state. A fault is an abnormal condition of hardware, software, or data that may cause a deviation from the expected sequence of behavior of some part of the system. An error state is a system state that can lead to a failure attributed to some aspect of that state. Finally, a failure occurs when a system does not meet its specifications; it is an externally observable event.

##### 4.2.2.2 Classes of Faults

There are a number of dimensions by which faults can be characterized; among them are the following:

- a. Duration: transient versus permanent
- b. Extent: local versus global
- c. Value: determinate versus indeterminate
- d. Criticality: flight-critical versus mission-critical versus mission-degrading

However, during the succeeding discussion, we shall take criticality as the prime dimension distinguishing three classes of faults from each other, that is, flight-critical versus mission-critical versus mission-degrading. Examples of functions in which faults would be in the three classes, respectively, are in-flight performance monitoring, ordinance management, and flight recording. Of course, it is possible for a fault to affect both flight-critical and mission-critical functions, for example, a fault in one or more of the executive functions, such as interrupt handling or synchronization.



#### 4.2.2.3 Techniques of Fault Tolerance

There are six areas of fault tolerance that must be addressed: (1) prevention, (2) masking, (3) detection, (4) diagnosis, (5) isolation, and (6) recovery.

4.2.2.3.1 Fault Prevention. Included in this area would be the practice of such software techniques as top-down design, structured programming, careful debugging, and the use of a higher order language. In hardware the designer should ideally use reliable components, rigorous quality control, nonvolatile memories, and careful design to minimize the occurrence of faults.

The major trade-off decisions to be made by a designer in the area of fault prevention are degree of prevention and availability versus cost, size, weight, processing time, and memory requirement.

4.2.2.3.2 Fault Masking. Fault masking uses redundancy to assure that the effect of a fault is completely contained within a system module. If sufficient redundancy exists in the module, which may be either hardware or software, the fault is concealed within the module; and no effect is propagated outside the module.

A key issue in masking is the size of the module within which masking occurs. In hardware, the masked module can vary from a few components to an entire subsystem or processor. Likewise, in software, redundancy can be built into a relatively small module or into a large module.

The trade-off design decisions to be made in this area are the same as in fault prevention.

4.2.2.3.3 Fault Detection. Fault detection is the starting point of most fault recovery implementations. It can be accomplished by either hardware or software methods, which can be grouped according to the time of their application:

- a. Initial testing, which takes place prior to normal use and serves to identify fault hardware elements

- b. Concurrent (on-line) detection, which takes place simultaneously with normal operation of the system
- c. Scheduled (off-line) detection, which takes place when normal operation is temporarily suspended
- d. Redundancy testing, which verifies that the various redundancy features of a system are ready to use when needed

Issues to be decided by the designer in the area of fault detection are the following: What percentage of faults is to be detected? With what level of confidence? Is fault detection to be accomplished by hardware, software, or combination? Which of the four above types of fault detection will be utilized?

Naturally, the greater the level of fault detection and the greater the level of confidence, the more elaborate are the methods, whether hardware or software, which must be used to settle these matters. As the complexity of fault detection rises, the availability of the system will rise (up to a point); but so will cost, size, weight, processing time, and memory requirements. Beyond a certain level of fault detection, the hardware and/or software required is so elaborate that there is more chance of a fault occurring in the detection hardware and/or software than in the equipment being tested.

4.2.2.3.4 Fault Diagnosis. Fault diagnosis is concerned with determining the specific nature of a fault which has occurred in order to repair it. On-line or off-line diagnosis during flight is not very practical for many avionic processing systems because repairs cannot be made easily during a mission. On ship or ground or on large aircraft, such as the P-3C, fault diagnosis might be more practical.

4.2.2.3.5 Fault Isolation. Once a fault has been detected and localized to a specific hardware or software module, the failed module must be isolated from the rest of the system to avoid propagating erroneous information throughout the system. Issues which the designer must decide are how large the modules should be (for example, should a hardware module, for the purposes of fault isolation, be a processor or a section of a processor) and when a module should be isolated (after the first detection of failure or after several attempts to get the module to respond).

4.2.2.3.6 Fault Recovery. Fault recovery comprises all actions taken after a fault has been detected and the failed unit has been isolated to restore some degree of normal operation in the system. Recovery algorithms can be divided into automatic and manually controlled schemes. Manual schemes are usually not too feasible for avionic use, except possibly for large aircraft such as the P-3C, because flight personnel do not have the time or knowledge to re-configure a processing system and because of space and weight constraints in the plane.

Automatic schemes can be subdivided into three classes: full recovery, partial recovery, and safe shutdown. Full recovery means that the system is restored to the same capability that existed before the fault. Failed hardware is replaced by spares; damaged programs and data are returned to a state existing before the fault. Partial recovery (also known as degraded recovery, graceful degradation, or fail-soft operation) returns the system to a fault-free state but with reduced capability. This means that some hardware modules have been eliminated from the system without replacement, some functions and/or data have been lost, or some functions now take longer or are performed less frequently than scheduled. Safe shutdown (also called fail-safe operation) is the limiting case of partial recovery. It is used when the remaining processing capability is below a minimum acceptable threshold. The goals of safe shutdown are to avoid damage to the remaining good programs, data, and hardware; to cease interactions with other systems and/or operators; and to deliver shutdown messages and diagnostic information to designated users.

Hardware-controlled systems use dedicated hardware to detect the presence of a fault and to initiate recovery, while software-controlled systems use special programs to accomplish this.

The principal advantage of hardware-controlled recovery systems lies in their independence of the operation of software immediately after the fault has occurred. Recovery control is transferred to software only after the software's ability to operate has been assured. The major disadvantage is that special hardware must be designed, built, and maintained.

The major advantage of software-controlled recovery systems is that existing hardware modules can be used, with resultant increased reliability and lower life-cycle cost. The major limitation is that the recovery must remain operational during the occurrence of a fault, since recovery cannot occur unless the recovery software can run correctly.

#### 4.2.2.4 Detection and Recovery Goals

The following is a list of goals deemed appropriate for the BASIC processing architecture study:

- a. 100-percent detection of all faults with a 100-percent confidence level
- b. Isolation of all failed units from the system
- c. Complete recovery from fault(s) in one or two flight-critical functions
- d. Complete recovery from fault(s) in one mission-essential function
- e. Partial recovery to degraded mode of operation from faults of two mission-essential functions
- f. Operation in degraded mode when fault(s) occur in mission-degradable functions, with loss of only those functions

#### 4.2.2.5 Design Criteria

The criteria used in making decisions on the choice of fault tolerance schemes are the following:

- a. Availability (including fault recovery goals)
- b. Life-cycle cost
- c. Use of standard, off-the-shelf hardware
- d. Number of processors and processing time required
- e. Memory storage requirements
- f. Bus requirements (especially data transfer rates)
- g. Implementation (hardware/software/firmware)
- h. Reliability (both hardware and software)
- i. Control (hardware or software)



- j. Degree of recovery and functions involved
- k. Diagnostic test in hardware, software, or firmware
- l. Type of fault detection

#### 4.2.3 Recovery Techniques

As described in paragraph 4.2.2, recovery from a fault must be preceded by detection of the fault and isolation of the fault to a system component so that the rest of the system is not affected. After these actions, recovery may be obtained through the use of redundant hardware or software, or through the operation of the system in a degraded mode. The objective of this paragraph is to recommend the best techniques for the implementation of these recovery functions in the BASIC information processing system, considering the constraints imposed by the state-of-the-art and the use of standard hardware and software.

##### 4.2.3.1 Detection/Isolation Techniques

Detection of faults and their isolation to a particular physical module is accomplished by performing a test on the module and comparing the module output against previously defined values. This comparison can indicate whether the module is functioning properly and could also be used to determine which submodule is malfunctioning or which group of further tests should be performed to isolate the fault to a particular submodule. These comparisons can be made by either the module under test (self-test) or some other module. If self-test is used, care must be taken that the comparison mechanism is working properly before more complex tests are made.

As has been described in paragraph 4.2.2, there are four types of tests that are appropriate for the testing of avionic functions. They are described in detail below.

4.2.3.1.1 Initialization Testing. Initialization testing is performed when the avionic system is first powered up prior to takeoff. This testing may also be performed as a confidence check after a system reinitialization following a

failure. Since this testing time is not critical and the resources to be tested are not required for other functions at the time of initialization, initialization testing should be the most complete of all the types of testing; all other tests should be a subset of the initialization tests. These tests must be sequenced in such a manner that very simple tests on very basic functions are tested first. With these basic functions verified, more complex tests can be made on higher level functions. This process can continue until very complex system level tests are made with confidence that the testing mechanism itself is in good working order. In addition, in a system like BASIC, these tests may be performed using an external computer and other test equipment which are not part of the simulated avionic system. This can provide a significantly greater level of confidence in the testing than can be attained simply by using the operational equipment.

4.2.3.1.1.1 Detection Capability/Confidence Level. The initialization test should be the most complete test and should check for all faults. All flight-critical failures should be detected with a 100-percent confidence level. For initialization testing, isolation of a fault in a particular module with a great degree of precision is not necessary, but detection of any fault that might jeopardize the continuation of the mission is mandatory. In addition, all mission-critical faults should be detected with a high degree (approximately 95 percent) of confidence, since time and money would be wasted by taking off and not being able to perform the intended mission.

4.2.3.1.1.2 Hardware Test Functions. Certain of the initialization tests must be implemented in hardware/firmware, since they must be performed in order to verify the proper execution of software. These tests must check the ability of a processor to read and write memory and to communicate to the outside world through at least one I/O channel, through which further software may be loaded. This involves the self-check of many data paths and control mechanisms which are best tested through the use of the special microcode incorporated into

the firmware of the computer. This section of microcode is entered automatically on power-on condition, and it includes a capability of automatic loading of memory from an I/O channel after the self-check functions are complete.

4.2.3.1.1.3 Software Test Functions. After software is loaded into the computer memory, a complete diagnostic test program may be run. This normally starts with a test of the individual instructions and ends with a test of the interrupt structure. At this point, the capability to transmit and receive data from external I/O channels can be tested. In the case of the MIL-STD-1553 bus, this capability could be attained by having the computer attempt to send a prespecified test message to itself.

After each computer has completed the above tests, the intercomputer and computer-peripheral communication paths may be tested. After this, system status may be determined and reported; this step completes the initialization testing.

4.2.3.1.2 On-Line Detection. On-line detection is performed concurrently with the execution of application processes. It normally checks the validity of data passing from one computer to another or from one segment of the computer to another. It also can be used to determine if a particular event has taken place within a specific time period. Normal implementation of on-line detection involves hardware parity checking and time-out interrupts. Time-out interrupts occur when an application process does not reset a test timer within a prespecified interval. Thus, if a process is in an infinite loop because of some hardware or software failure, a built-in test will indicate a failure.

4.2.3.1.2.1 Detection Capability/Confidence Level. Since on-line detection is basically a built-in-test concept, the detection capability and confidence levels for it depend on the state of the art and the cost/weight/performance penalties that the computer designer is willing to accept. For the most part, especially with standard hardware, these factors are beyond the control of the system designer. Most military computers have on-line fault detection capability in

compliance with the requirements of AR-10. Fault isolation capability is usually limited to an indication of a fault in the entire computer rather than in a particular module.

#### 4.2.3.2 Isolation Techniques

Once a fault is detected and isolated to a single physical module or set of modules, the system must be protected from the effect of that fault by the isolation of the faulty module from the system. For the purposes of this discussion, assume that a fault has been detected and isolated to one of the following units; the central processing unit, memory module, I/O channel, or I/O processor.

4.2.3.2.1 Central Processing Unit (CPU) Faults. If only one CPU exists in the computer and that CPU or any of its interfaces to the rest of the computer fails, then the only feasible approach to isolation is to disable the entire computer and remove the CPU from any of the interface buses with which it communicates. In a well-designed computer, this may be easily accomplished by removing power from the computer. This removal of power can be triggered by a self-test signal or by a signal from an operator or an external hardware monitor.

In a computer which has multiple CPU's, such as a multiprocessor, or which has an independent I/O processor, it may be possible to disable only the faulty CPU and have the remainder of the computer act as a uniprocessor. This capability would have to be designed into the hardware and activated either by a self-test output or an external signal

4.2.3.2.2 Memory Module Faults. Faulty memory modules can be isolated from the remainder of the system if some type of virtual memory translation hardware is incorporated into the computer. If redundant memory is available, this can be substituted for the faulty physical memory module by appropriate changes to the virtual-to-physical address translation tables. If redundant memory is not available, it may be possible, through memory protection mechanisms, to cause an interrupt if an attempt is made to reference the faulty memory module.



4.2.3.2.3 I/O Channel Failure. If an I/O channel fails, all communication between that channel and the CPU or memory must be inhibited. This inhibition can be accomplished by changing the address of the channel to an unused I/O channel address or by using a hardware mechanism that essentially switches the failed channel off and replaces it with a redundant channel.

4.2.3.2.4 I/O Processor (IOP) Failure. If an I/O processor fails in a computer which has a CPU capable of performing I/O and CPU tasks in a time-shared mode (as in the case for the AN/AYK-14), then the IOP may be shut down via a mode command and the computer will still be able to perform, although with reduced performance. If the CPU cannot perform I/O tasks, then the entire computer would have to be shut down.

4.2.3.3 Recovery Techniques. Once a faulty module has been isolated from the system, steps must be taken to recover the functional capability that had been impaired by the fault. If this is not possible, then the system must be reconfigured such that the loss of that function does not cause other functions to fail.

4.2.3.3.1 Full Recovery. If a fault in hardware is to be completely masked, then redundant hardware or unused time resources must be employed. This process may be implemented with redundant computers in the system (which may be used for testing in a normal, fault-tree system) and possibly with unused memory and I/O modules in the operational computers. If a CPU fails and removal of a computer from the system is necessary, then its replacement must be a computer that has access to all the programs executed by the faulty computer, and all the constant and global data used by it. These programs and constant data could be prestored in the redundant computer or stored in a mass memory to which the backup computer has access. Global data could be obtained from other processing elements. The replacement computer must have access to all the peripherals and other system processing elements with which the original computer communicated. Under these conditions, full recovery is possible. The faulty computer is shut down. The replacement is loaded with the appropriate

programs and data, is initialized, and brought into operation at the beginning of the next major cycle. Thus, processing capability and data are lost for only a short time, at least from the operator's viewpoint.

Because a memory module may fail, an extra memory module in the computer is necessary; it may be substituted for the failed module by changing the virtual-to-physical address translation tables. If the failed module contained programs or constant data, the replacement must be reloaded with this information from an external source, on the data must have been prestored in the redundant module.

If an I/O channel fails, complete recovery requires that an identical, redundant channel, connected to the same peripherals, be available. In this case, switching channels is normally a simple matter.

If an I/O processor fails, a redundant IOP must be available within the computer, along with a hardware mechanism to switch out the failed IOP and replace it with the redundant module. This is not normally the case in most computers, with the result that complete recovery from an IOP failure must be handled by replacement of the entire computer, as if the CPU had failed.

4.2.3.3.2 Partial Recovery (Graceful Degradation). Partial recovery usually means that full recovery is not possible because the necessary redundant resources are not available. In this case, the computer which has suffered the failure is shut down, and its tasks are assigned as extra duties to another operational computer. This backup computer must have access to the same peripherals and system elements as the original. Since the backup computer must now perform extra processing, more time will be required to execute all the processes. Since tasks are normally dispatched on a priority basis, some lower priority tasks may not be executed during some major cycles. If this condition is not acceptable, some tasks may be given variable priority, such that their dispatching priority increases every time they are not executed during a major cycle. Thus, graceful degradation for a processing system essentially means that lower priority tasks do not execute as often as the system specification would require. This

technique is clearly not acceptable for flight-critical functions; it is useful for operator display, navigation, and communication functions.

4.2.3.3.3 Safe Shutdown. Shutting down a function is necessary when a failure occurs in a module for which there is no backup or redundant hardware provided. This normally occurs when a mission sensor, such as a radar transmitter or a MAD receiver, fails. In this case, the scheduling procedure must be revised so that the tasks using these data are not executed. Periodic tasks that process the data to or from the failed sensor should not be executed, and demand tasks that require these data should be replaced with processes that return an indication that the required function has failed.

#### 4.2.4 Software Considerations

Several recent studies have indicated that software is the major contributor to overall system cost, and projections are that the cost of software will be 90 percent of system cost by 1985. These projections may very well prove true, since technological improvements in hardware, such as very large-scale integration (VLSI), have produced drastic decreases in the cost of hardware. Such new memory technologies as magnetic bubbles also give promise of substantially reducing the cost of memory.

Technological progress has also been made in software in recent years. New developments in software give some promise of keeping software costs under control, although not to the same extent as in hardware. Such developments as higher order languages, top-down design, structured programming, and modular programming are examples of these.

##### 4.2.4.1 Higher Order Languages

The advantages of using a higher order language (HOL) for application DU's implemented in software are:

- a. Shorter programming time
- b. Shorter debugging time

- c. Lower programming cost
- d. Increased programming transparency (that is, the ability to write programs without need for detailed knowledge of the system)
- e. Increased portability of software (that is, the ability to use software on several different hardware systems with only minor changes required)

The disadvantages are that the HOL may result in a less efficient code with more processing time and more memory required. However, the advantages of using an HOL seem to outweigh the disadvantages, especially if a standard HOL is used.

#### 4.2.4.2 Top-Down Design

The approach being used to design the BASIC laboratory information processing system is a top-down methodology in which the designer starts with the application requirements and works down from there through decomposition, and so forth. This methodology is somewhat different from the current approaches in which functions are assigned for implementation in software at the beginning of software design. Instead, the top-down methodology does not choose a medium for implementation until well along in the design process. However, this methodology does support top-down design of software; once the decision to implement a particular DU in software is made, a top-down design of the software for that DU can easily be attained, with all of the functional elements (DE's) known from previous steps.

#### 4.2.4.3 Structural/Modular Programming

The advantages of structural/modular programming are the following:

- a. Supports and facilitates top-down design
- b. Allows proof of correctness. The only way to be certain that a software program functions properly is by establishing a "proof of correctness." Debugging only shows the presence of errors and not their absence and, hence, is insufficient. In general, a large software program is not amendable to such a proof, but a smaller module is.
- c. Reduces programming cost. Figure 4-2, taken from a General Electric brochure, shows that programming cost varies exponentially with size of program. Therefore, breaking software into modules should reduce costs.



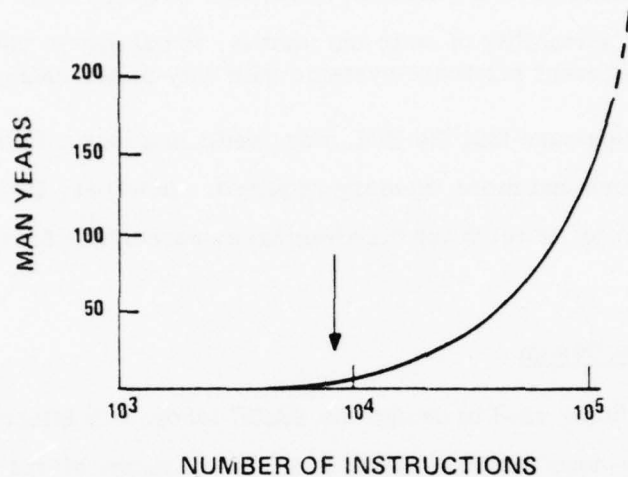


Figure 4-2. Software Size/Cost

- d. Facilitates software management, design, maintenance, and verification by easier reading of code
- e. Allows use of standard or common modules resulting in increased cost effectiveness
- f. Allows flexibility in assignment of tasks to processors, in recovery after a fault, and in changing the system for various applications
- g. Facilitates system growth by allowing new functions to be added relatively easily
- h. Promotes portability of software
- i. Meshes well with the NAVAIRDEVCON/Univac Design Methodology

The possible disadvantages of structured/modular programming are that more initial programming time may be required, and that the code generated may demand a slightly longer processing time and/or slightly more memory. However, the advantages more than outweigh the disadvantages.

#### 4.2.5 Compatibility with Hardware/Software Standards

##### 4.2.5.1 Current Applicable Directive and Standards

Among the current directives applicable to the design and development of computer processing systems are DoD Directives 5000.29, which covers the management of software, 5000.31, which specifies the acceptable higher order languages (HOL's), and 5000.xx (draft), which specifies the approved computer architectures. In addition, a memorandum of 30 March 1977 from the Secretary of the Navy specifies that the AN/AYK-14 shall be the Navy standard airborne computer. The AN/AYK-14 computer includes the AN/UYK-20 instruction set and emulates the latter's architecture.

The current standard Navy HOL's are the CMS-2 for data processing and SPL-1 for signal processing, with DoD-1 to be the future DoD standard HOL. Other existing standards specify the MIL-STD-1553, -1553A, and -1553B buses as the standard Navy avionic buses. No standard microprocessor has yet been chosen, nor is there a policy for standardization in the areas of computer resident memories or mass memory storage.

##### 4.2.5.2 Discussion

Implementation of DU's in hardware and software should be in compliance with existing DoD and Navy directives and standards unless there is some compelling reason to request a waiver. The purpose of such documents are to minimize the proliferation of hardware, increase the portability of software, reduce logistic requirements, and reduce life-cycle costs.

The disadvantage of standardization is the difficulty of incorporating the latest technological advances (hardware and software) when processor architecture, language, bus protocols, and so forth are fixed. In Naval avionic applications, however, the advantages of standardization over an "equipment generation" outweigh the disadvantages; therefore, standardization will be followed to the maximum extent possible.

## APPLICATION BINDING

## 5.1 INTRODUCTION

The previous sections have defined a set of application requirements, broken those requirements into decompositional elements (DE's), combined the DE's into decompositional units (DU's), and defined a baseline structure within which the DU's can execute their functions. The next step in the architectural development is the selection of implementation media for the DU's and the composition of these implementations into an overall system. This section will develop implementations for the DU's, propose a BASIC system architecture, and present the rationale for the selections recommended. Specific areas to be discussed are:

- Information Flow — a description of data flow among DU's based on the S-3A operational program
- DU Functional Grouping — combining DU's into functional groups for distribution among processing nodes for the BASIC application
- Message Structure — organizing information flow and defining messages which conform to the MIL-STD-1553 bus protocol for the BASIC application
- Hardware Configuration — hardware required to implement the processing nodes
- Processing Requirements — software execution parameters corresponding to the various DU's to be simulated in the BASIC application
- Control Structure — control mechanisms necessary for control of the DU's and a recommended implementation for the control mechanisms.

An analytical approach was employed in the development of this preliminary architecture. This implies that a simulation remains to be performed via the Generalized Computer System Simulator (GCSS) so that the architecture recommended herein can be verified in terms of interactions between the data transfers

and the processes that use the data. A GCSS simulation is currently under development; the results will be described in a future supplement to this report.

## 5.2 DECOMPOSITIONAL UNIT IMPLEMENTATIONS

Having defined the DU's, it now becomes necessary to select an implementation medium for each DU in a manner which will optimize the assessment criteria parameters defined in Section 4. In all cases, a software implementation was chosen. The rationale for this choice with respect to the assessment criteria is presented in the following paragraphs.

### 5.2.1 Flexibility

DU's implemented in hardware or firmware are considerably more difficult to change than DU's implemented in software. In addition, software DU's can take advantage of the rapid increases in hardware performance which accompany processor technology growth. Thus, replacement of a computer (which executes many DU's) with a higher performance computer automatically increases the performance of those DU's.

### 5.2.2 Fault Tolerance/Recovery

It is assumed that software errors will be minimized by proper programming practices, simulation, bench testing, and flight testing. In the proposed architecture, failures in the computing hardware on which the DU's execute do not result in loss of the DU function, since the software may be executed (perhaps in a degraded mode) in a different computer. This functional redundancy of DU's would not be possible in hardware implementation, since it would result in unacceptable weight and cost penalties in redundant hardware.

### 5.2.3 Software Considerations

The DU's can be programmed in HOL and, thus, take advantage of the inherent cost savings in a modular, structured programming approach. Since the



DU's are a result of a top-down partitioning methodology, their specifications and interfaces are known in a format highly compatible with a computer program performance specification. Indeed, the software module specifications are very nearly a by-product of the methodology process.

#### 5.2.4 Compatibility With Software/Hardware Standards

A DU implementation in software can be made completely compatible with existing standards. No unique or new hardware need be procured, and no non-standard languages need be used.

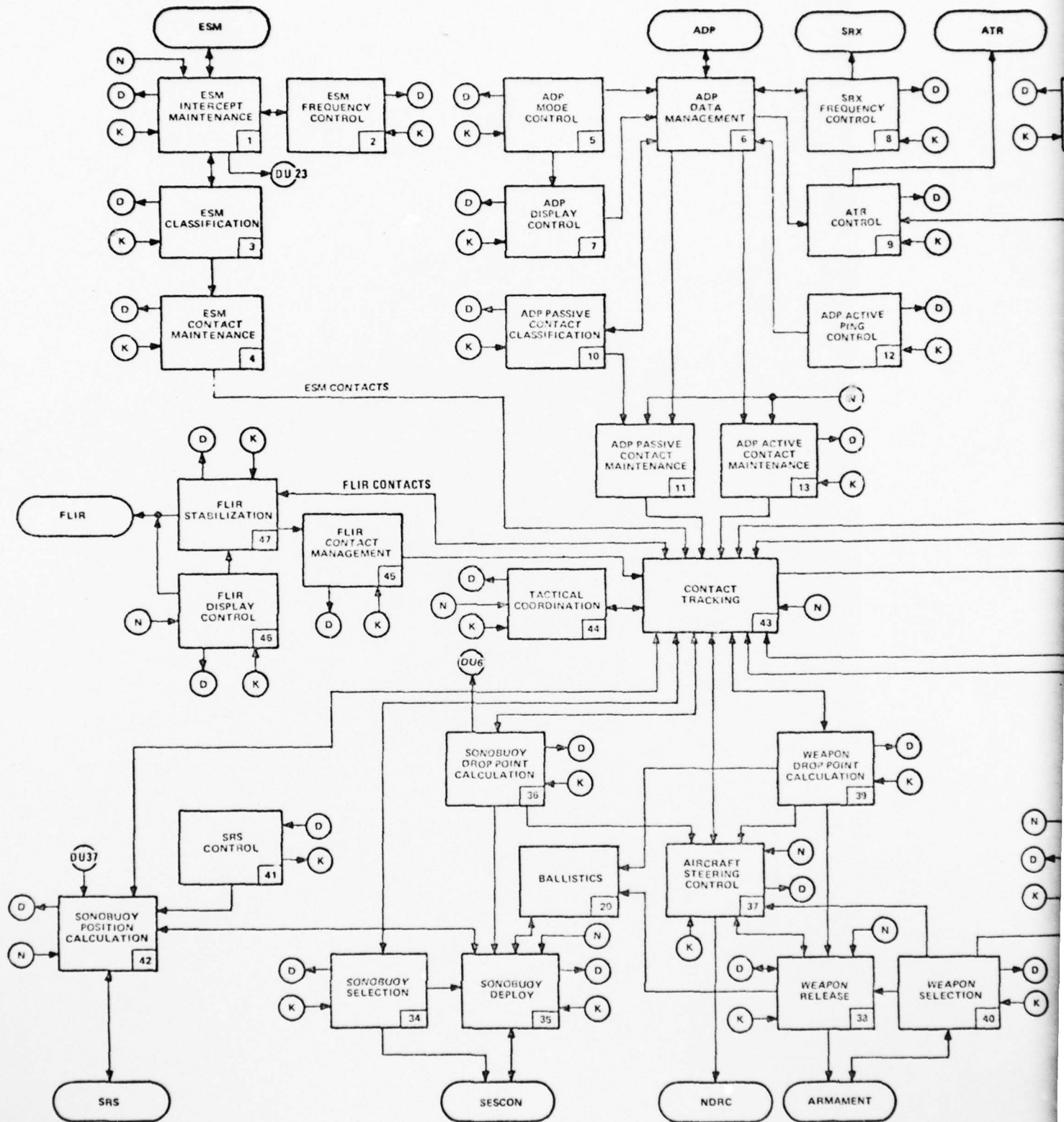
### 5.3 DECOMPOSITIONAL UNIT FUNCTIONAL GROUPING

In deriving a multiple processing system architecture, it is necessary to organize the DU's shown in Figure 5-1 into operational groups with functions related according to a set of assessment criteria so that they can be distributed among several processing elements in an optimal fashion. However, lacking weighting parameters and associated physical assessment criteria, the design decisions for grouping DU's in the recent S-3A architecture study were essentially based on minimizing I/O rates between functional groupings, while not exceeding the processing capability of a moderate-power, microprocessor-based minicomputer.

A DU grouping proposed as a result of the study described in Reference 4 is shown in Table 5-1. In addition to the operational DU's shown, allowances are made for executive, test, and diagnostic functions. The executive functions for each processor differ depending on the scheduling requirements dictated by the particular DU group. The executive function overhead is currently projected to be on the order of approximately 30 percent. The overhead for the test and diagnostic functions is currently estimated as being less than 10 percent. The structure of the executive is defined in more detail in paragraph 5.5.

TABLE 5-1. PROPOSED DU GROUPING

DU NO.	FUNCTION	PROCESSOR GROUP							
		1	2	3	4	5	6	7	8
1	ESM Intercept Maintenance	X							
2	ESM Frequency Control	X							
3	ESM Classification	X							
4	ESM Contact Maintenance	X							
5	ADP Mode Control				X				
6	ADP Data Management				X				
7	ADP Display Control				X				
8	SRX Frequency Control				X				
9	ATR Control				X				
10	ADP Passive Contact Classification					X			
11	ADP Passive Contact Maintenance					X			
12	ADP Active Ping Control						X		
13	ADP Active Contact Maintenance						X		
14	MAD Display Control			X					
15	MAD Data Management			X					
16	MAD Signal Feature Recognition			X					
17	MAD Contact Maintenance			X					
18	RADAR Display Control		X						
19	RADAR Data Management		X						
20	Ballistics	X							
21	RADAR Contact Maintenance	X							
22	Display Operator Control								X
23	Tactical Display Management								X
24	Display Refresh Management								X
25	Keyset Control								X
26	NAV Position Control							X	
27	NAV Position Calculation							X	
28	NAV Data Management							X	
29	NAV Track History							X	
30	COMM Receive		X						
31	COMM Control		X						
32	COMM Data Management		X						
33	COMM Transmit		X						
34	Sonobuoy Selection	X							
35	Sonobuoy Release	X							
36	Sonobuoy Drop-Point Calculation	X							
37	Aircraft Steering Control							X	
38	Weapon Release	X							
39	Weapon Drop-Point Calculation	X							
40	Weapon Selection	X							
41	SRS Control								X
42	Sonobuoy Position Calculation								X
43	Contact Tracking							X	
44	Tactical Coordination							X	
45	FLIR Contact Maintenance	X							
46	FLIR Display Control	X							
47	FLIR Stabilization	X							
--	Executive	EXEC	EXEC	EXEC	EXEC	EXEC	EXEC	EXEC	EXEC
--	Test/Diagnostic	T/D	T/D	T/D	T/D	T/D	T/D	T/D	T/D



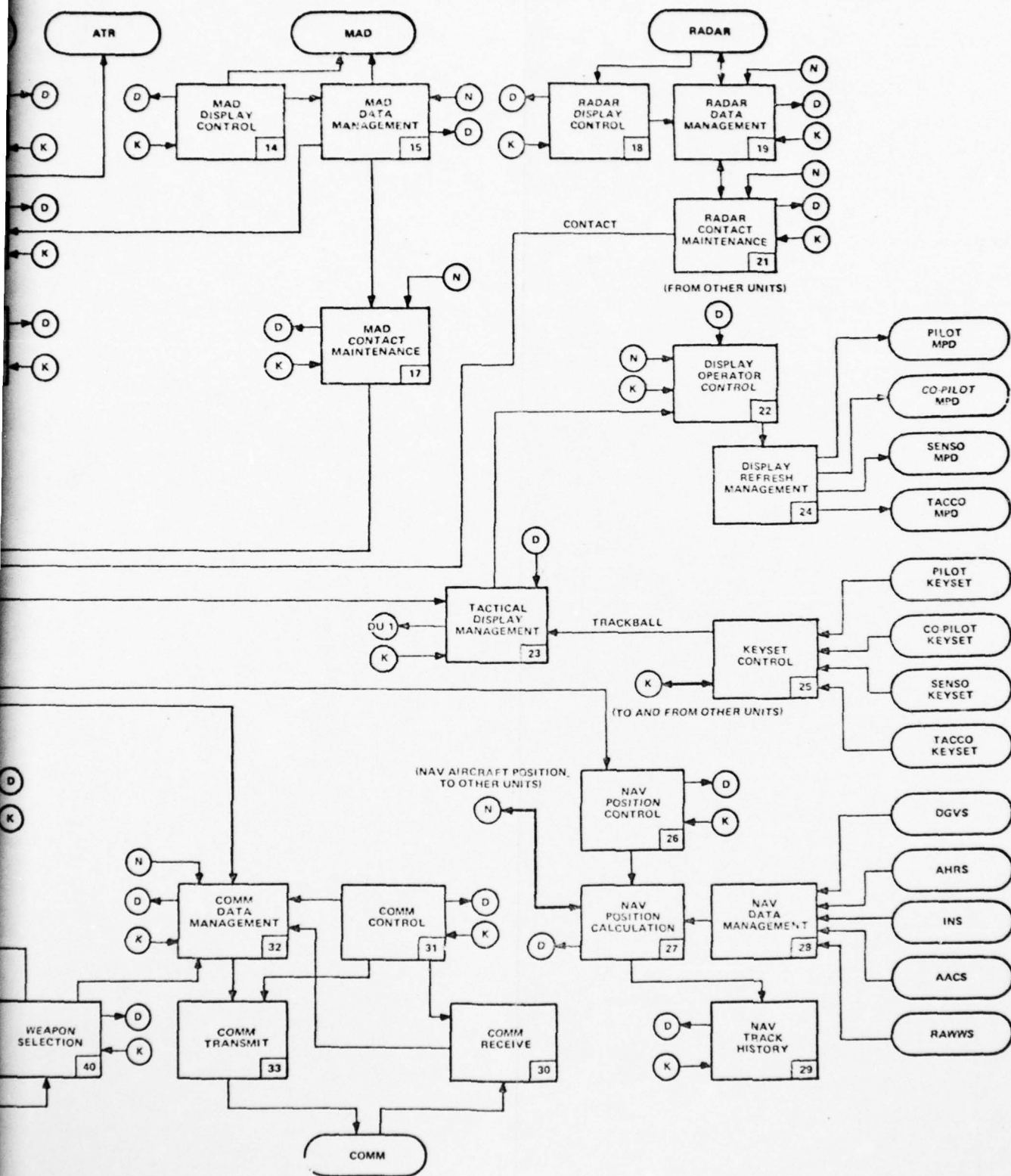


Figure 5-1. Decompositional Unit Information Flow Diagram



#### 5.4 DECOMPOSITIONAL UNIT/SUBSYSTEM INFORMATION FLOW

The partitioning of the S-3A operational program into 47 DU's was developed in Section 3 and is summarized in Table 3-4 along with the associated sensors/subsystems. Figure 5-1 is an excerpt from Reference 3 which shows directional data flow paths among these DU's and sensors/subsystems. The symbols contained in the information flow diagram of Figure 5-1 are interpreted as follows:

- Boxes represent individual DU's whose general functions are described in paragraphs 3.2.1.2.1 through 3.2.1.2.47 of Section 3.
- Oblong circles represent elements external to the information processing system, that is, sensors/subsystems.
- Circles with the letter K indicate data obtained from the Keyset control element, DU25.
- Circles with the letter D indicate display data (alerts, cues, and so forth) to be transferred to the display operator control element, DU22.
- Circles with the letter N indicate navigation data (aircraft position, speed, course, and so forth) obtained from the NAV position calculation element, DU27. As indicated previously, each DU consists of a number of DE's. These DE's are the next lower level of decomposition which are the smallest executable entities (tasks). However, at this time the S-3A partitioning at the DU level appears to provide sufficient understanding and visibility regarding data flow and element interactions for establishing baseline requirements for the BASIC avionic information processing system architecture.

#### 5.5 DU/DU COMMUNICATION

Application processes will be distributed among various processing elements in a multiple processor, organized information processing system. This distribution will require the sharing of data among application processes and is referred to as communication.

##### 5.5.1 Modes of Communication

For purpose of discussion, a process can be considered to be either a data source or destination. If the data source and destination processes are both associated with the same processor, directly accessing the same memory, then the transfer of data will be referred to as intraprocessor communication. If the data source and destination processes are associated with different processors, each

with its own private memory, then message passing is required to move the data to the destination process. Data transfer employing message passing will be referred to as interprocessor communication.

#### 5.5.2 Address Translation Mechanism

In order to relieve application processes from the concern of actual data locations, a communication scheme could be employed in which address translation is performed by a separate mechanism. This scheme will be referred to as an address translation mechanism. The general operation of the proposed scheme is depicted in Figure 5-2. As shown, data locations are preassigned to specific groups and, from the point of view of the application process, are always directly accessible. In operation, when the application process proceeds to fetch data, these data are intercepted by the address translation mechanism. This mechanism can consist of a series of status registers, for each group of data, which contain flags indicating the current location of the desired data. When employing the intraprocessor communication mode, the data are local to the process, and the mechanism immediately proceeds to read the data. When employing the interprocessor communication mode, data are remote to the process, and an I/O mechanism is invoked to fetch a copy of the data. After the data are fetched, the status is changed to local, the processor is notified, and the process continues. A possible implementation of the mechanism could invoke an interrupt-service routine to initiate the necessary I/O procedures. Each status register associated with a data group could point to a location in a table which would contain the appropriate service routine. This table could be defined at system generation time with provisions for modification to accommodate system reconfiguration.

#### 5.6 DATA/MESSAGE MAPPING

The interaction of data among the functions of the processor groups shown in Table 5-1 requires the use of an interconnection/bus structure for data transfer. As indicated above, two types of data transfers will be considered: intraprocessor and interprocessor communications. The common bus structure that

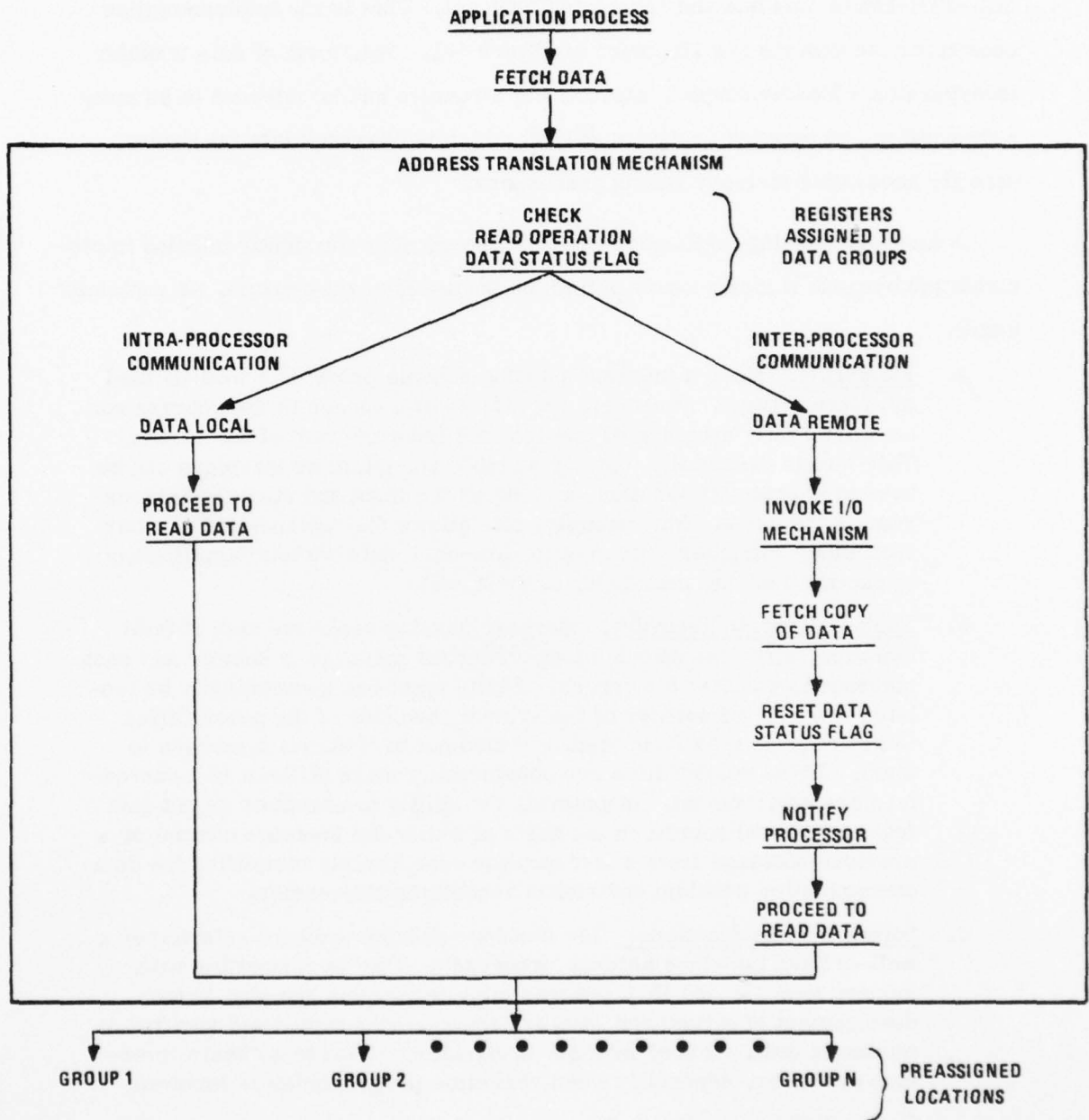


Figure 5-2. Communication Scheme

will be considered in the subsequent paragraphs will be the serial, multiplexed MIL-STD-1553A data bus and associated protocol. This is the implementation chosen for the generic bus structure of Figure 4-1. This form of data transfer incorporates a loosely coupled interconnect structure and is referred to as message passing, as opposed to tight coupling, which is characterized by shared, directly accessible memory among processors.

A message passing implementation was chosen over the tightly coupled implementation because it more nearly optimizes the assessment criteria, as explained below.

- a. Flexibility. Since a message passing scheme provides a well-defined interface between processes, the effects of a change in the process can be more easily ascertained and isolated from the rest of the system. This means essentially that any suitable algorithm or technique can be used to execute the function, as long as the input and output messages remain the same. For example, this allows the replacement of older technology computers with newer, low-cost units without significantly impacting upon the remainder of the system.
- b. Fault Tolerance/Recovery. Message passing eases the task of fault isolation, since the source of an erroneous message is known, and each message is checked for errors. Faulty units can automatically be isolated from the remainder of the system, because of the possibilities to refuse messages from them and also not to transmit messages to them. These possibilities are considerably more difficult in a shared-memory environment. In general, the ability to accept or reject data from a selected source on the basis of either the message content or a previous message from a test monitor considerably simplifies the data contamination problem and makes reconfiguration easier.
- c. Software Considerations. The message passing protocol establishes a well-defined interface between processes. This is compatible with modern modular and HOL programming techniques and aids in the development of structured software which can be tested and verified at minimum cost. Shared memory environments lead to software interface problems, especially when real-time programming is involved.
- d. Compatibility with Standards. The implementation chosen is the standard (MIL-STD-1553) serial bus and appears to meet the data transfer requirements for interprocessor communications.

#### 5.6.1 Data Transfer Rates

Figure 5-3 provides a summary of data transfers required for the current S-3A operational program. The circles indicate the complement of DU's associated



AD-A072 427

NAVAL AIR DEVELOPMENT CENTER WARMINSTER PA COMMUNICA--ETC F/G 5/2  
BASIC LABORATORY ARCHITECTURE PLAN.(U)

MAY 79 S GREENBERG, C JOECKEL, R MEJZAK

IDWA-45-78-04

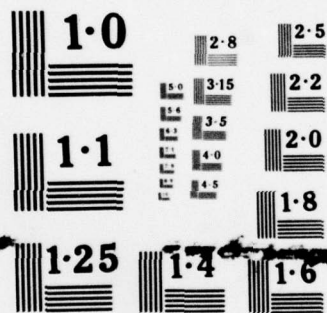
UNCLASSIFIED

NADC-79161-40

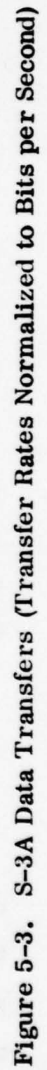
NL

2 OF 4  
AD  
A072427





NATIONAL BUREAU OF STANDARDS  
MICROCOPY RESOLUTION TEST CHART



with each of the processor groups (1 through 8) as established in Table 5-1. An additional processor group (PG-9) performs the system health monitoring function over the other processor groups, acts as the bus controller, and controls reconfiguration (if it is not itself the failed unit.) The oblong circles indicate the sensors/subsystems and their relationship to the processor groups. Note that the data transfer rates are normalized to bits per second and, therefore, only characterize the data flow on a gross basis. The self-loops shown imply intraprocessor communications. All other data transfers shown are considered to be interprocessor communications. Since the data transfers include only the basic information, a data transfer analysis should also take into account the growth and overhead factors indicated in Figure 5-3.

#### 5.6.2 MIL-STD-1553A Bus Considerations

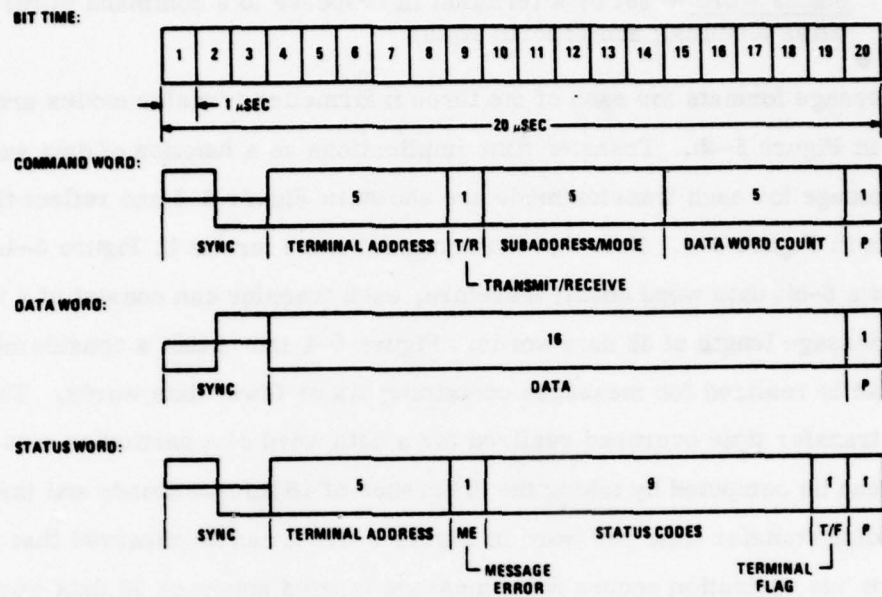
The 1553A data bus permits a maximum of 32 terminals to be connected on any one bus. Information is transferred on a single shielded, twisted pair line at a 1-megahertz bit rate. Control of information transmission and reception on the bus resides with the bus controller, which initiates all transfers. The data bus may employ three modes of information transfer:

- Controller-to-terminal
- Terminal-to-controller
- Terminal-to-terminal

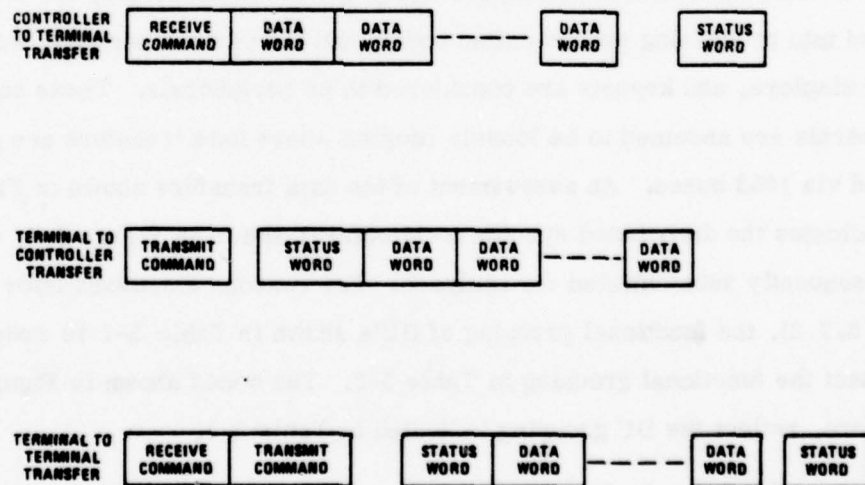
A terminal is considered to be the electronic unit necessary to interface the bus with the subsystem and the subsystem with the bus. A redundant bus controller, when not functioning as a controller, may operate as a terminal. Data are transferred serially in 20-microsecond frames, each divided into 17 bit times of 1 microsecond and one 3-microsecond sync interval. All messages are addressed by use of three types of words, as shown in Figure 5-4a:

- Command Word — sent by bus controller to the appropriate terminal; specifies message type; and sets data word count for subsequent transfer
- Data Word — contains 16 bits of message data, sync pattern, and a parity bit





A. WORD FORMATS



\*TIME GAP

B. MESSAGE FORMATS

Figure 5-4. MIL-STD-1553A Formats

- Status Word — set by a terminal in response to a command word; identifies terminal; and reports status

Message formats for each of the three information transfer modes are shown in Figure 5-4b. Transfer time implications as a function of data words per message for each transfer mode are shown in Figure 5-5 and reflect the formats in Figure 5-4. Note that the command word format in Figure 5-4a consists of a 5-bit data word count; therefore, each transfer can consist of a maximum message length of 32 data words. Figure 5-4 shows that a considerable overhead is realized for messages containing six or fewer data words. The actual transfer time overhead realized for a data word of a particular message length can be computed by taking the difference of 16 microseconds and the corresponding transfer time per word in Figure 5-5. It can be observed that more efficient bus utilization occurs when message lengths approach 32 data words. This is further demonstrated in Figure 5-6, which characterizes bus efficiency for various message lengths.

## 5.7 SYSTEM CONFIGURATION ANALYSIS

In developing an information processing system architecture, the DU's are mapped into processing groups called nodes; and the various sensors, subsystems, displays, and keysets are considered to be peripherals. These nodes and peripherals are assumed to be loosely coupled where data transfers are performed via 1553 buses. An assessment of the data transfers shown in Figure 5-3 indicates the distributed system architecture, shown in Figure 5-7, which is subsequently substantiated via analyses. For reasons discussed later (paragraph 5.7.2), the functional grouping of DU's shown in Table 5-1 is redefined to reflect the functional grouping in Table 5-2. The nodes shown in Figure 5-7, therefore, reflect the DU grouping indicated in Table 5-2.

### 5.7.1 Data Transfer Analysis

An analysis is performed based on the data transfer requirements shown in Figure 5-2 and the system configuration shown in Figure 5-7. As indicated previously, only interprocessor data transfers are considered, that is, those

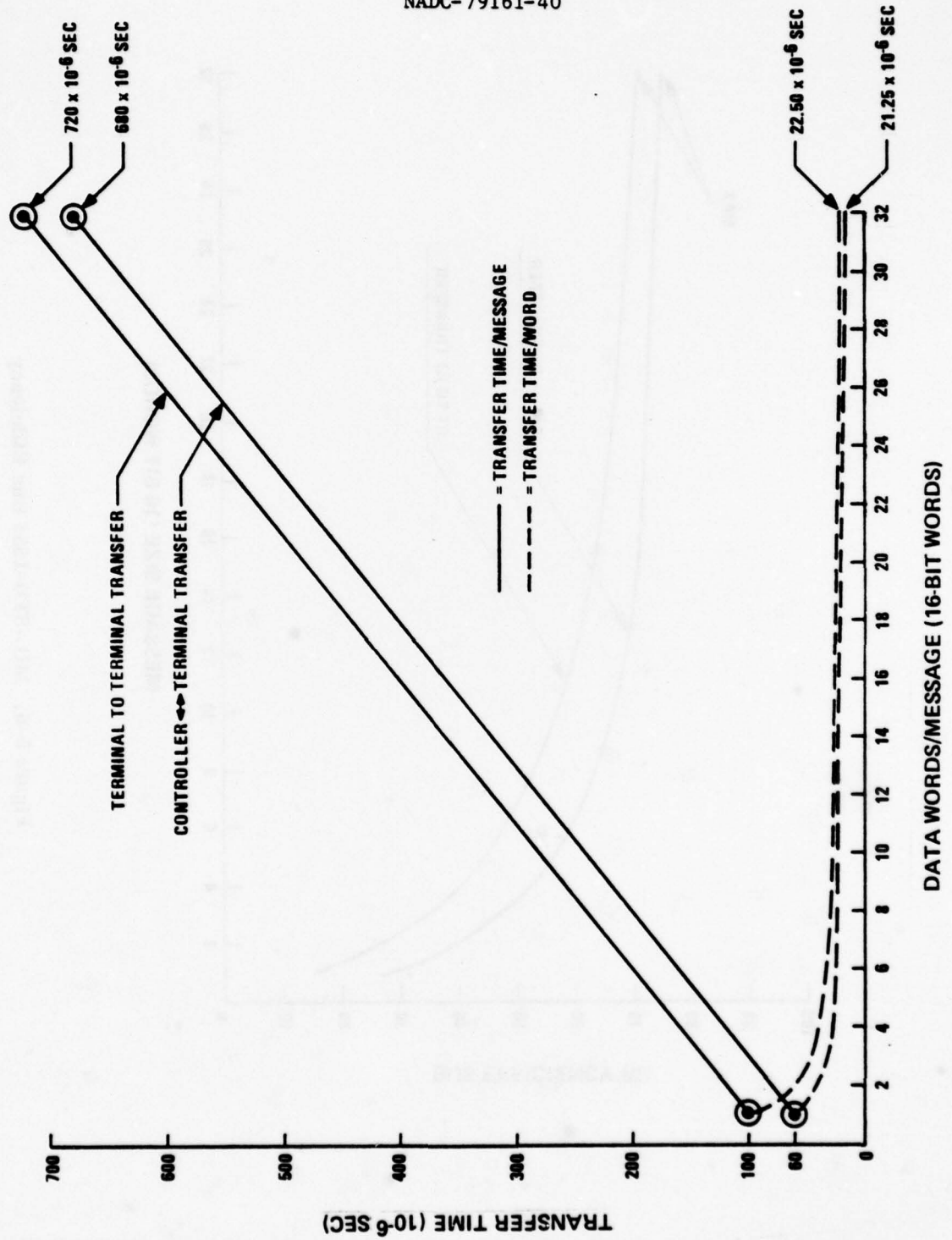


Figure 5-5. MIL-STD-1553 Bus Message Transfer Times

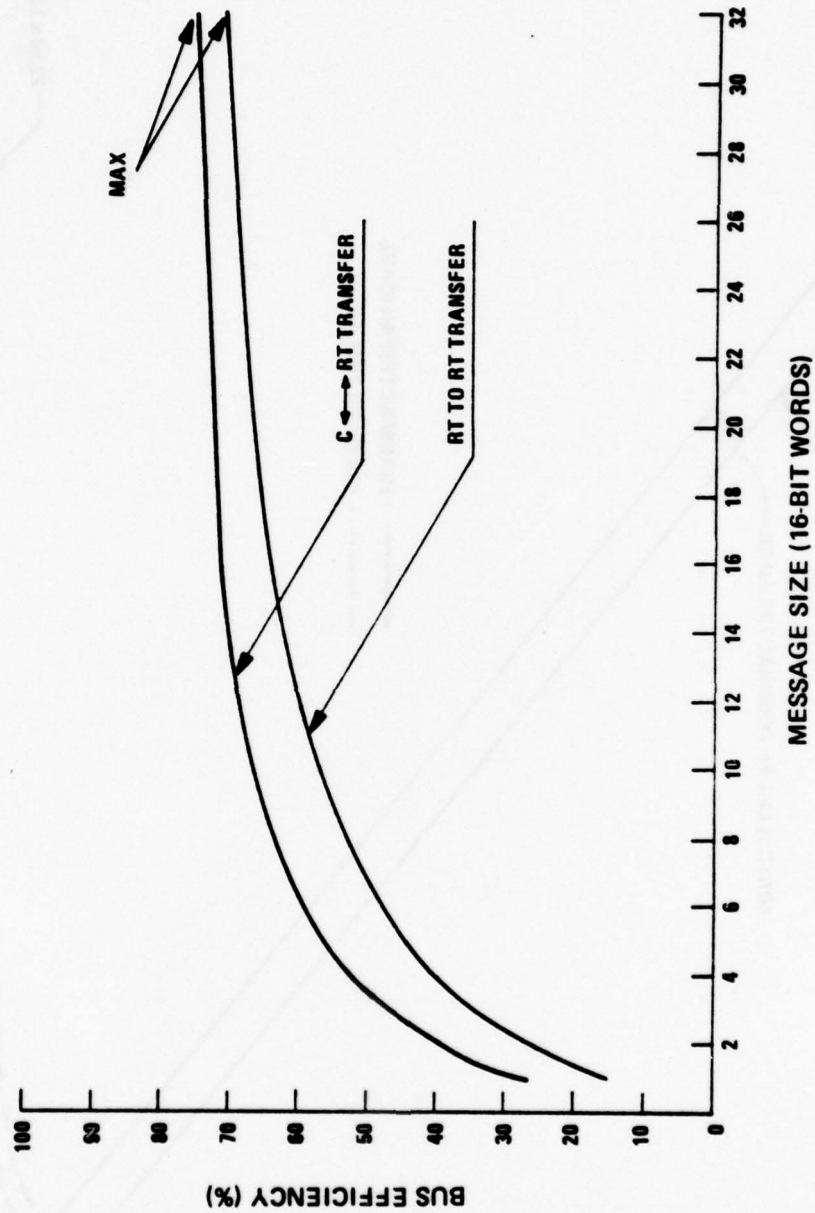


Figure 5-6. MIL-STD-1553 Bus Efficiency



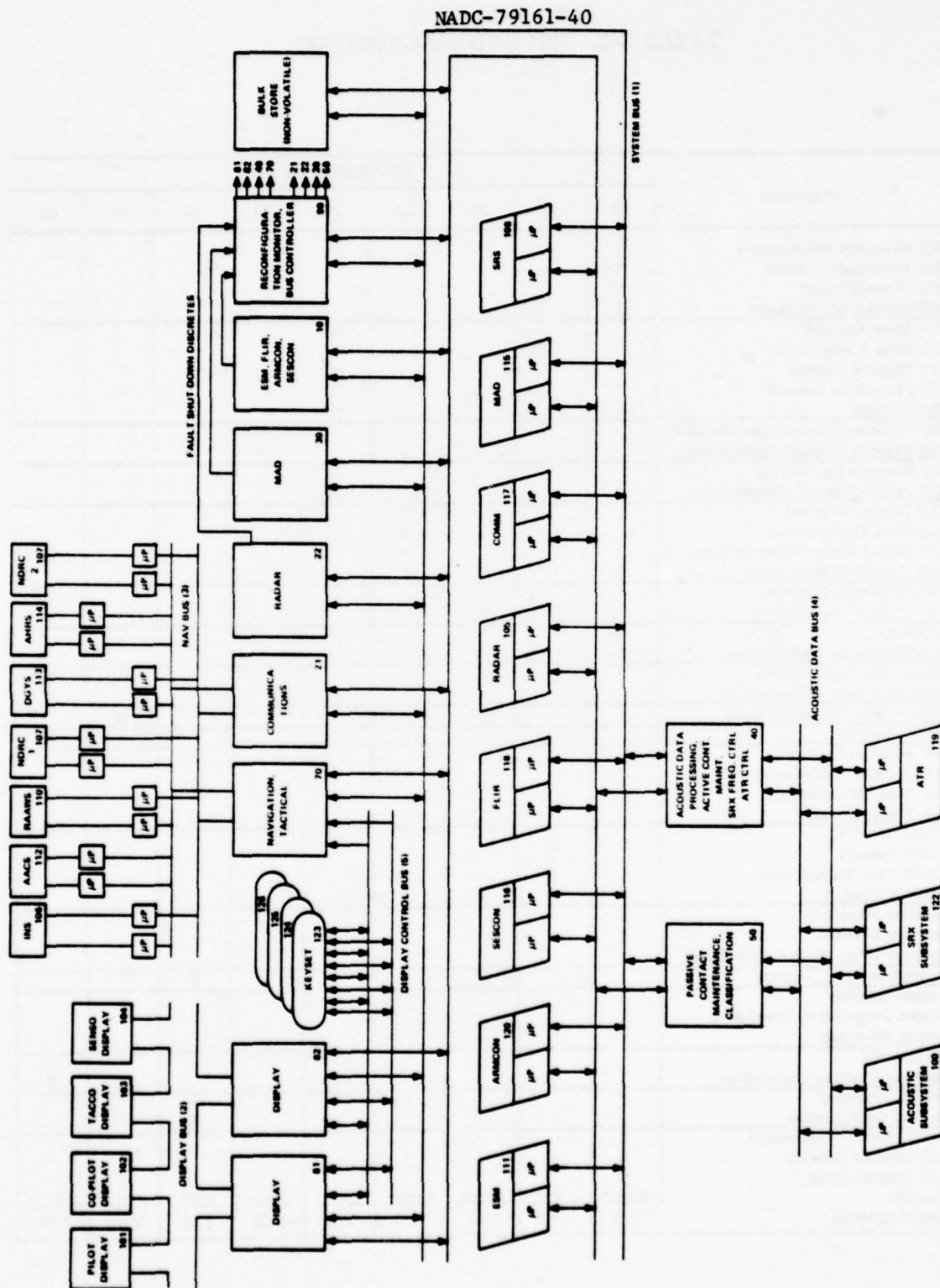


Figure 5-7. BASIC Architecture

NADC-79161-40  
TABLE 5-2. REVISED DU GROUPING

DU NO.	FUNCTION	PROCESSOR GROUP									
		10	21	22	30	40	50	70	81	82	
1	ESM Intercept Maintenance	X									
2	ESM Frequency Control	X									
3	ESM Classification	X									
4	ESM Contact Maintenance	X									
5	ADP Mode Control					X					
6	ADP Data Management					X					
7	ADP Display Control					X					
8	SRX Frequency Control					X					
9	ATR Control					X					
10	ADP Passive Contact Classification						X				
11	ADP Passive Contact Maintenance						X				
12	ADP Active Ping Control					X					
13	ADP Active Contact Maintenance					X					
14	MAD Display Control				X						
15	MAD Data Management				X						
16	MAD Signal Feature Recognition				X						
17	MAD Contact Maintenance				X						
18	RADAR Display Control			X							
19	RADAR Data Management			X							
20	Ballistics	X									
21	RADAR Contact Maintenance	X									
22	Display Operator Control									X	
23	Tactical Display Management									X	
24	Display Refresh Management								X		
25	Keyset Control									X	
26	NAV Position Control							X			
27	NAV Position Calculation							X			
28	NAV Data Management							X			
29	NAV Track History							X			
30	COMM Receive		X								
31	COMM Control		X								
32	COMM Data Management		X								
33	COMM Transmit		X								
34	Sonobuoy Selection	X									
35	Sonobuoy Release	X									
36	Sonobuoy Drop-Point Calculation	X									
37	Aircraft Steering Control							X			
38	Weapon Release	X									
39	Weapon Drop-Point Calculation	X									
40	Weapon Selection	X									
41	SRS Control									X	
42	Sonobuoy Position Calculation									X	
43	Contact Tracking							X			
44	Tactical Coordination							X			
45	FLIR Contact Maintenance	X									
46	FLIR Display Control	X									
47	FLIR Stabilization	X									
--	Executive	EXEC	EXEC	EXEC	EXEC	EXEC	EXEC	EXEC	EXEC	EXEC	
--	Test/Diagnostic	T/D	T/D	T/D	T/D	T/D	T/D	T/D	T/D	T/D	

data transfers performed by means of 1553 buses. This implies that the communication structure among nodes and peripherals will conform to the protocol requirements of the 1553 bus. The 1553 word and message formats to be employed are shown in Figure 5-5.

#### 5.7.1.1 Message Formats

For purposes of analysis, all nodes and peripherals are considered to be remote terminals, with the exception of the node designated as the controller (Node 90). All data transfers are assumed to be initiated by the controller. When a terminal is transmitting a message, that terminal is considered to be the source of the transmission. When a terminal is receiving a message, that terminal is considered to be the destination. Source and destination terminals are identified by means of terminal addresses that are specified by the controller in the command word. Since terminal addresses are not yet determined for the terminals (nodes and peripherals) in Figure 5-7, the numerical assignments shown can be considered to be pseudo-addresses. Other information contained in the command word includes a data word count which permits messages of variable length to be transmitted with up to a maximum of 32 16-bit words. In addition, a transmit/receive (T/R) bit indicates whether the addressed terminal is to transmit or receive the message.

#### 5.7.1.2 Message Scheduling

Messages to be transmitted are scheduled during predetermined periods called cycles. The shortest scheduling period is called a minor cycle; all other cycles are an integer number of minor cycles. Figure 5-8 shows scheduling periods corresponding to various cycles in which a minor cycle is defined as 50 milliseconds. The cycles repeats after a time period equal to a major cycle. The number of minor cycles in a major cycle is some power of 2 (for example, 8, 16, 32). Figure 5-8 provides an example employing this message scheduling scheme. As shown therein, six subaddresses are employed to perform the required transfer for the example.



1. SCHEDULE 4 TRANSFERS WITH 32 DATA WORDS/MESSAGE EACH, EVERY 50 MSEC (CYCLE=0)  $\rightarrow$   $4 \times 20 \times 32 = 2560$  WDS/SEC
2. SCHEDULE 1 TRANSFER WITH 32 DATA WORDS/MESSAGE EVERY 100 MSEC (CYCLE = 1)  $\rightarrow$   $1 \times 10 \times 32 = 320$  WDS/SEC
3. SCHEDULE 1 TRANSFER WITH 24 DATA WORDS/MESSAGE EVERY 200 MSEC (CYCLE = 2)  $\rightarrow$   $1 \times 5 \times 24 = 120$  WDS/SEC

$\Sigma = 3000 \text{ WDS/SEC}$

IT MAY BE NOTED THAT 6 SUBADDRESSES ARE REQUIRED, I.E., 4 FOR THE 50 MSEC, 1 FOR THE 100 MSEC, AND 1 FOR THE 200 MSEC TRANSFERS

**Figure 5-8. Message Scheduling**



### 5.7.1.3 Message Structure

In order for the S-3A information flow requirements to adapt to the system configuration of Figure 5-7, messages are constructed to perform the 1553 bus data transfers. In constructing these messages, flexibility objectives were considered in terms of minimizing both bus utilization and subaddresses so that future growth could be accommodated. The results of an analysis indicate that a minor cycle of 50 milliseconds is required to realize the required data transfers. A complete list of messages required for the system configuration of Figure 5-7 is shown in Table 5-3 and reflects data transfer rates as determined from References 2 and 3. As indicated previously, the source and destination terminal assignments are pseudo-addresses which can be reassigned prior to implementation in the BASIC laboratory. However, command words can be constructed by employing the word counts, transmitter subaddresses, and receiver subaddresses directly from Table 5-3. Note that the command word for the transmitter would have the T/R bit set to 1, and the receiver would have the T/R bit set to 0. Scheduling information can also be obtained directly from the cycle field indicated. The maximum cycle required is 4, which corresponds to 800 milliseconds, as shown in Figure 5-8. Asynchronous messages in Table 5-3 correspond to those messages with a 0 entry in the period field and no entry in the cycle field. In the case of asynchronous messages, both the transmitter and receiver subaddresses are assigned a value of 30 for compatibility with either the 1553A or 1553B bus. Other information provided in Table 5-3 includes the identification of the bus (corresponding to Figure 5-7) used for each message transfer. Provisions are also made for monitoring the health of the individual modes by means of the controller/monitor node (90). This is accomplished by requiring each node to issue a message to be received and evaluated by the monitor every minor cycle (50 milliseconds).

TABLE 5-3. MESSAGE STRUCTURE (Sheet 1 of 3)

	SRC NDE	XMT SUB ADR	DST NDE	WORDS PER SEC	PERIOD IN MSEC	BUS ID	CYC CODE	WRD CNT	RCV SUB ADR	TIME IN μ SEC
***										
*	10	1	70	640	500	1	0	32	1	725
*	10	2	82	100	450	1	2	23	1	545
*	10	3	82	600	450	1	0	30	2	685
*	10	4	82	450	1000	1	1	30	3	685
*		5					1	16	4	405
*	10	6	111	100	1000	1	2	20	1	485
*	10	7	118	240	1000	1	1	24	1	565
*	10	29	90	160	50	1	0	8	1	245
*	10	30	70	18	0	1		3	30	145
*	10	30	82	30	0	1		3	30	145
*	10	30	116	6	0	1		3	30	145
*	21	1	10	10	1000	1	4	10	1	285
*	21	2	70	3000	2000	3	0	32	2	725
*		3					0	32	3	725
*		4					0	32	4	725
*		5					0	32	5	725
*		6					1	32	6	725
*		7					2	16	7	405
*	21	8	117	150	1000	1	2	30	1	685
*	21	29	90	160	50	1	0	8	2	245
*	21	30	82	6	0	1		3	30	145
*	22		105	3	0	1				
*	22	1	10	150	1500	1	2	30	2	685
*	22	2	82	50	50	1	0	3	5	145
*	22	3	105	60	1000	1	3	30	1	685
*	22	29	90	160	50	1	0	8	3	245
*	22	30	10	3	0	1		3	30	145
*	22	30	105	3	0	1		3	30	145
*	30	1	82	400	40	1	0	20	6	485
*	30	2	82	3600	160	1	0	32	7	725
*		3					0	32	8	725
*		4					0	32	9	725
*		5					0	32	10	725
*		6					0	32	11	725
*		7					0	32	12	725
*	30	29	90	160	50	1	0	8	4	245
*	30	30	40	3	0	1		3	30	145
*	30	30	70	3	0	1		3	30	145
*	30	30	82	3	0	1		3	30	145
*	30	30	115	3	0	1		3	30	145
*	40	1	50	300	100	1	1	30	1	685
*	40	2	82	530	1000	1	1	32	13	725
*		3					1	21	14	505
*	40	4	100	900	100	4	0	30	1	685
*		5					4	30	2	685
*	40	6	119	10	1000	4	4	10	1	285
*	40	7	122	10	1000	4	4	10	1	285
*	40	8	70	10	1000	1	4	10	5	285
*	40	9	82	10	1000	1	4	10	15	285
*	40	29	90	160	50	1	J	8	5	245
*	40	30	10	3	0	1		3	30	145
*	40	30	82	3	0	1		3	30	145
*	50	1	40	100	100	1	1	10	1	285
*	50	2	70	800	2000	1	0	32	3	725

TABLE 5-3. MESSAGE STRUCTURE (Sheet 2 of 3)

SRC NDE	XMT SUB ADR	DST NDE	WORDS PER SEC	PERIOD IN MSEC	BUS ID	CYC CODE	WRD CNT	RCV SUB ADR	TIME IN μ SEC
***									
*	3					2	32	4	725
*	50	29	90	160	50	0	8	6	245
*	50	30	40	3	0	1	3	30	145
*	50	30	82	3	0	1	3	30	145
*	60	30	50	0	1		3	30	145
*	70	1	10	300	50	0	15	3	385
*	70	2	10	420	50	0	21	6	505
*	70	3	21	210	100	1	21	1	505
*	70	4	21	960	1000	1	32	23	725
*	70	5	22	210	100	1	21	1	505
*	70	6	30	300	50	1	15	1	385
*	70	7	30	420	50	0	21	2	505
*	70	8	40	105	200	2	21	2	505
*	70	9	50	105	200	2	21	2	505
*	70	10	82	1240	50	0	19	16	485
*		11				0	21	17	505
*		12				0	21	18	505
*	70	13	82	1500	2000	5	30	19	685
*		14				0	30	20	685
*		15				0	30	21	685
*	70	16	82	320	1000	5	32	22	725
*	70	17	82	100	2000	5	11	23	305
*	70	29	90	160	50	0	8	5	245
*	70	30	10	13	0	1	13	30	345
*	70	30	10	63	0	1	21	30	505
*	70	30	82	16	0	5	16	30	405
*	81		101	12000	25	2			
*	81		102	36000	25	2			
*	81		103	40000	25	2			
*	81		104	40000	25	2			
*	81	29	90	160	50	0	8	8	245
*	82	1	10	240	50	0	12	1	325
*	82	2	22	20	50	0	1	2	105
*	82	3	40	40	50	0	2	3	125
*	82	5	81	4000	500	0	32	1	725
*		6				0	32	2	725
*		7				0	32	3	725
*		8				0	32	4	725
*		9				0	32	5	725
*		10				0	32	6	725
*		11				1	16	7	405
*	82	12	108	8	1000	4	8	1	245
*	82	29	90	160	50	0	8	9	245
*	82	30	10	48	0	1	3	30	145
*	82	30	21	6	0	1	3	30	145
*	82	30	22	3	0	1	3	30	145
*	82	30	30	3	0	1	3	30	145
*	82	30	40	9	0	1	3	30	145
*	82	30	40	6	0	1	3	30	145
*	82	30	50	3	0	1	3	30	145
*	82	30	70	25	0	1	5	30	185
*	82	30	70	14	0	1	14	30	365
*	90	1	21	160	50	0	8	29	245
*	90	2	30	160	50	0	8	29	245

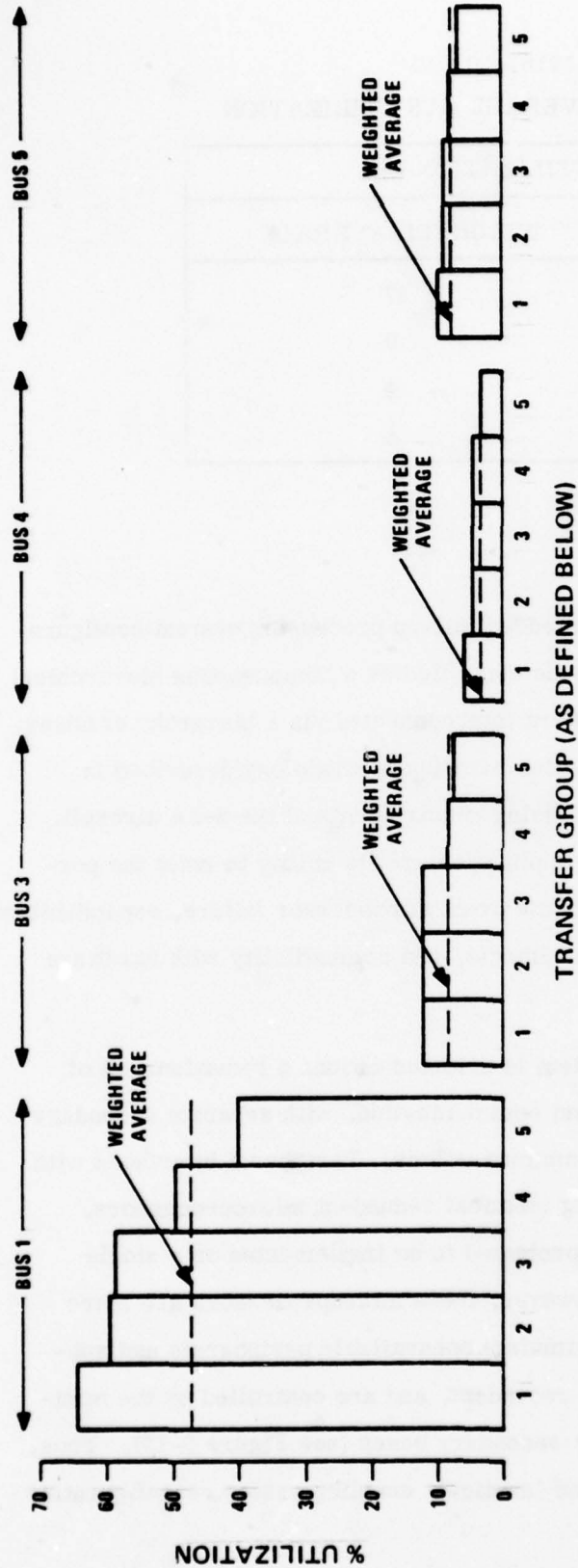
TABLE 5-3. MESSAGE STRUCTURE (Sheet 3 of 3)

	SRC NDE	XMT SUB ADR	DST NDE	WORDS PER SEC	PERIOD IN MSEC	BUS ID	CYC CODE	WRD CNT	RCV SUB ADR	TIME IN μ SEC
***										
*	90	30	21		0	1		7	30	225
*	100	9	40	900	100	4	0	32	4	725
*		9					1	26	5	605
*	105	14	22	60	50	1	0	3	3	145
*	106	15	70	240	50	3	0	12	8	325
*	107	16	70	120	200	3	2	24	9	565
*	108	17	82	8	1000	1	4	8	24	245
*	110	19	70	10	200	3	2	2	10	125
*	111	20	10	100	250	1	2	25	5	585
*	112	21	70	20	200	3	2	4	11	165
*	113	22	70	60	200	3	2	12	12	325
*	114	23	70	180	50	3	0	8	13	245
*	115	24	30	48	50	1	0	3	2	145
*	117	26	21	150	0	1		3	30	145
*	123	1	82	3	0	5		3	30	145
*	124	1	82	40	0	5		20	30	485
*	125	1	82	40	0	5		20	30	485
*	126	1	82	40	0	5		20	30	485

5.7.1.4 Bus Utilization

The bus utilization/transfer times shown for each message in Table 5-3 are a function of word length and correspond to the terminal-to-terminal transfers characterized in Figure 5-5. In addition, since a gap time of 2 to 5 microseconds occurs for each message transfer, 5 microseconds are included with each message transfer time to reflect a worst-case situation. As can be observed from Figure 5-8, a bus is not uniformly loaded over an entire scheduling interval of 800 milliseconds; that is, a scheduling interval consists of the 16 50-microsecond scheduling periods P1 through P16, after which the 800-millisecond interval repeats. A maximum or peak bus loading occurs when all cycles (0 through 4) are coincident as depicted for scheduling period P1 in Figure 5-8. Figure 5-9 characterizes utilization of the various 1553 bus structures shown in Figure 5-7 with respect to transfer groups as defined therein. A summary of bus utilization for the peak loading period P1 and the weighted average for an 800-millisecond interval is given in Table 5-4.





TRANSFER GROUP	APPLICABLE PERIOD (SEE FIGURE 5-7)	APPLICABLE CYCLE (SEE FIGURE 5-8)					NUMBER OF OCCURRENCES IN P <sub>1</sub> - P <sub>16</sub> INTERVAL
		0	1	2	3	4	
1	P <sub>1</sub>	X	X	X	X	X	1
2	P <sub>9</sub>	X	X	X	X		1
3	P <sub>5</sub> , P <sub>13</sub>	X	X	X	X		2
4	P <sub>3</sub> , P <sub>7</sub> , P <sub>11</sub> , P <sub>15</sub>	X	X				4
5	P <sub>2</sub> , P <sub>4</sub> , P <sub>6</sub> , P <sub>8</sub> , P <sub>10</sub> , P <sub>12</sub> , P <sub>14</sub> , P <sub>16</sub>	X					8

Figure 5-9. MIL-STD-1553 Bus Utilization for Synchronous Messages

TABLE 5-4. PEAK/AVERAGE BUS UTILIZATION

BUS ID	UTILIZATION (%)	
	PEAK	WEIGHTED AVERAGE
1	64	47
3	12	9
4	7	4
5	9	8

### 5.7.2 Processing Analysis

#### 5.7.2.1 Introduction

Figure 5-7 represents the recommended full-up processing system configuration for the BASIC laboratory. It may be classified as a homogeneous hierarchical system, since it uses standard computers interconnected via a hierarchy of buses. The system is derived (via the information handling methodology described in Section 2 of this report) from the processing requirements of the S-3A aircraft. The primary requirements for BASIC application are its ability to meet the performance requirements, availability in the event of processor failure, expandability to accommodate new or changing requirements, and compatibility with hardware and software standards.

Toward these objectives, the system is oriented around a redundant set of buses used for subsystem-to-subsystem communication, with separate secondary buses used for computer/peripheral communications. Peripheral interfaces with a secondary bus are implemented using identical redundant microprocessors, which, in a platform application, are projected to be implemented on a single board. For the BASIC application, however, these microprocessors are more powerful, flexible devices which can simulate nonavailable peripherals and subsystems. These microcomputers are redundant, and are controlled by the mini-computers attached to their particular secondary buses (see Figure 5-10). Thus, the interfaces can be made standard and identical, enabling easier reconfiguration and expansion.

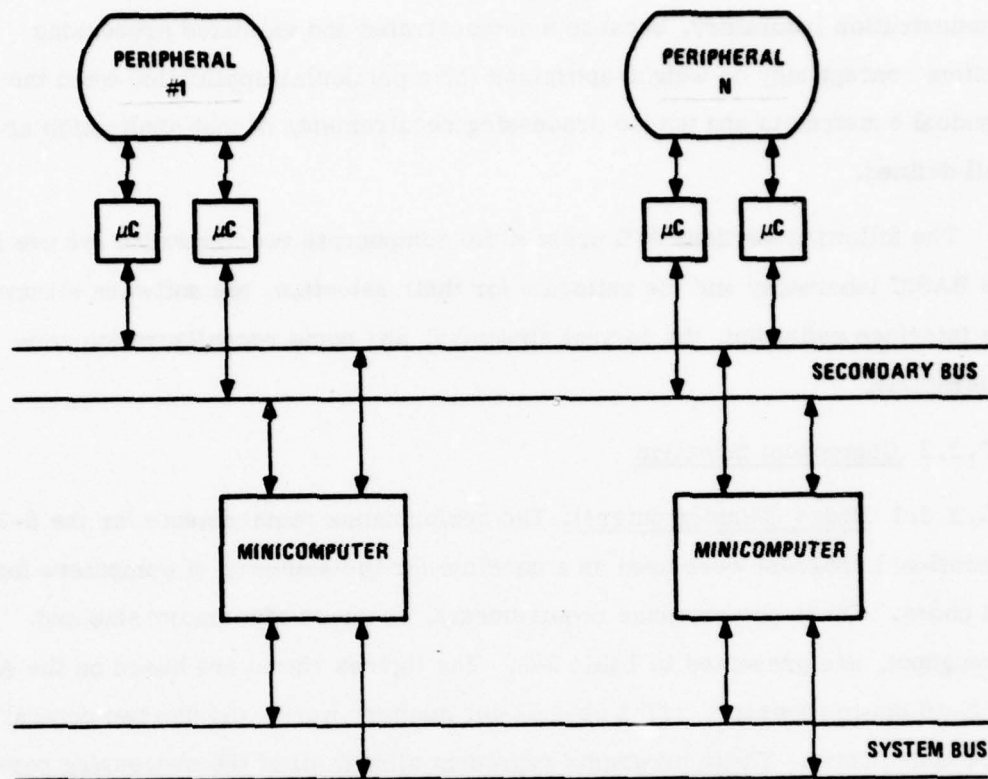


Figure 5-10. Secondary Bus Interfaces

Availability is provided by redundant hardware, redundant data paths, and reconfiguration software stored in the minicomputers and in a bulk storage medium accessible to all the processors. Additionally, one computer is dedicated to the test/monitor function and can be used as a spare in case of a processor failure.

The recommended architecture represents a system in which cost, flexibility, and availability are considered more important than weight and processor speed. Future requirements may be accommodated through the use of additional computers rather than through the use of excess processing capability within the existing computers. This approach reduces the cost of programming, since interaction between modules, especially for the purpose of increasing real-time speed of response, is minimized. This approach is also cost-effective for a

demonstration laboratory, because a demonstrated and validated processing system concept may be weight-optimized for a particular application when the physical constraints and unique processing requirements of that application are well defined.

The following sections will present the components recommended for use in the BASIC laboratory and the rationale for their selection, the software structure, the interface definition, the control structure, and some reconfiguration considerations.

#### 5.7.2.2 Component Selection

5.7.2.2.1 Nodes (Minicomputers). The performance requirements for the S-3A operational program were used as a baseline for the selection of computers for the nodes. These performance requirements, in terms of memory size and throughput, are presented in Table 5-5. The figures shown are based on the AN/AYK-10 multiprocessor, which uses 32-bit memory words and has two central processing units. These programs represent almost all of the processing performed by the AN/AYK-10. The processing time for the executive is included in other portions of the operational program (and has proved impossible to accurately separate). The functional description of the subprograms listed in Table 5-4 can be found in paragraph 3.1.2 of this report.

Table 5-6 presents the estimated requirements for the various nodes based on the operational requirements of Table 5-5 plus considerations of availability and compatibility with ongoing technology programs. The figures in Table 5-6 are based on a 16-bit minicomputer, since the standard AN/AYK-14 would be more than adequate to meet the processing requirements. The configurations recommended are AN/AYK-14 minicomputers. Before the BASIC architecture is fully implemented, newer versions of the standard minicomputer will probably be available. The recommended configurations will be regarded, therefore, as



TABLE 5-5. SUBPROGRAM SYSTEM RESOURCE REQUIREMENTS

SUBPROGRAM	NUMBER OF INSTRUCTION WORDS	NUMBER OF LOCAL DATA WORDS	NUMBER OF DYNAMIC DATA WORDS
KEYPAC, DISPAC	16,800	5,000	350
NAV, STEER, and SLTA	8,400	1,200	500
COMM	21,100	1,200	160
ARMCON, SESCON, and Ballistics	6,000	400	100
ACCON, PACK, and ACTIVE	34,200	7,200	1,700
FLIR	1,600	200	50
RADAR	3,300	300	130
MAD	2,200	3,800	50
ESM	1,700	1,400	120

TABLE 5-6. PROCESSING NODE SIZING ESTIMATES FOR BASIC ARCHITECTURE

NODE	ESTIMATED NUMBER OF INSTRUCTIONS	ESTIMATED NUMBER OF DATA WORDS	ESTIMATED THROUGHPUT (OPERATIONS PER SEC)
10 - ESM, FLIR, ARMCON, SESCON	14,000	4,000	175K
21 - COMM	31,650	2,750	75K
22 - RADAR	5,000	860	30K
30 - MAD	3,300	7,600	225K
40 - ACCON, ADP, ACTIVE, SRX, ATR CTRL	20,000	10,000	290K
50 - Passive Contact Maintenance, Classification	14,000	8,000	206K
70 - NAV, Tactical	12,600	3,400	180K
81/82 - Display	36,000	60,000	400K

representing capability rather than as unique hardware devices. The recommended configurations for each node are described below.

#### Node 10

The Node 10 processor performs the search stores control, armament control, FLIR, and ESM functions of the S-3A. This node has modest memory, speed, and I/O requirements. Since loading will not be excessive, Node 10 could act as a backup processor for either Node 30 or the reconfiguration monitor.

The number of ESM signals processed could be increased from 16 to 32 with slight additional bus loading. A 32K-word memory module would be adequate for the operational requirements. The recommended configuration, however, contains facilities for two 64K modules, so that Node 10 can act as a backup for the reconfiguration monitor, the MAD processor, or both if necessary.

The I/O requirements are such that the primary bus is not unduly loaded by the approximately 200 words per second, transmitted between the ESM subsystem and Node 1. Since a single connection to the primary bus has obvious advantages in terms of communication and reconfigurability, this option was chosen.

An IOP option was not considered necessary for the operational requirements, but the role of backup for the reconfiguration monitor requires it. An extended arithmetic unit (EAU) would be desirable since ESM processing requires many arithmetic operations, and floating-point arithmetic is more compatible with high-level language programming than fixed-point arithmetic. The EAU is not strictly necessary for proper operation, but the recommended configuration should be able to accept an EAU if desired. For maximum flexibility, therefore, an XN-1 configuration chassis is recommended. A discrete interface module (DIM) is required for control of the processor from external sources. A summary of the recommended configuration is given in Table 5-7.

TABLE 5-7. NODE 10 CONFIGURATION

CHASSIS - XN-1		
MODULE	DESIGNATION	QUANTITY
Power Supply	PCM-2	(1)
Memory	32K word	(1)
Memory Control	MCM	(1)
General Processor	GPM	(1)
Serial Interface	SIM	(1)
Discrete Interface	DIM	(1)
I/O Processor	IOP	(1)

Nodes 21 and 22

Nodes 21 and 22 are represented to be two computers for several reasons.

- a. The requirements of the radar and communication subsystems are such that subsystem-unique processing will probably also occur in that node (that is, the Transmission and Information Exchange System (TIES) communication processing uses the AN/AYK-14 as a subsystem processor). This will require processing capability in excess of the S-3A requirements to accommodate this subsystem-unique processing.
- b. With separate nodes, the programming may be done independently with minimized interaction between the processes.
- c. With two physical nodes, the computers can act as backups for each other and for Node 70 in case of failure.

Notwithstanding the above discussions, implementing these two nodes in one AN/AYK-14 was considered cost-effective for the BASIC application. However, to retain the functional separation required above, the AN/AYK-14 for Nodes 21 and 22 should be assigned two terminal addresses; the memory used by the communications and radar functions should be distinct; and all I/O's between the communications and radar functions should be conducted over the primary bus. This will allow future separation of the node into two computers with minimum impact.

The processing requirements for Nodes 21 and 22 are such that an XN-2 configuration could be used. Three 32K memory modules or two 64K memory modules would satisfy the storage requirements and allow reconfiguration in case of failure. I/O requirements are modest, approximately 4000 words per second, indicating that this node need not contain an IOP. Since the processing for this node is primarily a management function and few extensive arithmetic operations are performed, an EAU was not considered necessary. Although an XN-2 chassis type could be used, an XN-1 is recommended because of its greater flexibility and expandability. A DIM is also necessary for external control. A summary of the recommended configuration of Nodes 21 and 22 is presented in Table 5-8.

#### Node 30

The MAD processing requirements for the S-3A can be reasonably well accommodated by an AN/AYK-14, XN-1 configuration using an EAU. As in the case of Node 1, an EAU is not strictly necessary, but programming complexity

TABLE 5-8. CONFIGURATION FOR NODES 21 AND 22

CHASSIS — XN-1		
MODULE	DESIGNATION	QUANTITY
Power Supply	PCM-2	(1)
Memory	32K word	(3)
	or	
	64K word	(2)
Memory Control	MCM	(1)
General Processor	GPM	(1)
Serial Interface	SIM	(4)
Discrete Interface	DIM	(1)



is significantly decreased if floating-point arithmetic can be used; and the speed requirements are such that the microprogrammed floating-point capability might not be fast enough for the application. Floating-point instructions in the AN/AYK-14 run at a 180K operations per second rate. The overall speed required for the MAD application is 220K operations per second. This speed requirement could probably be met, since the program does not contain only floating-point calculations; but careful programming would be required, and growth would be prohibited.

Memory requirements are small, and a single 32K memory module will more than suffice. An IOP is not considered necessary for this node since the I/O requirements are small. A summary of the recommended configuration is presented in Table 5-9.

#### Node 40 — Acoustic Data Processing

The processing requirements for Node 40 are somewhat difficult to assess since this node performs many different functions on a demand basis as required by the sensor operator (SENSO). Node 40 has one major periodic function, which is the input of acoustic data (12 milliseconds of processing time every 100 milliseconds). Other tasks are in response to operator commands or to sonobuoy

TABLE 5-9. NODE 30 CONFIGURATION

CHASSIS — XN-1		
MODULE	DESIGNATION	QUANTITY
Power Supply	PCM-2	(1)
Memory	32K word	(1)
Memory Control	MCM	(1)
Serial Interface	SIM	(1)
Extended Arithmetic Unit	EAU	(1)
Discrete Interface	DIM	(1)

signals. Although these other tasks occur infrequently, they are randomly spaced in time and can thus cause heavy peak loading. The processing capability must therefore be adequate to ensure proper response time under high-load conditions, even though the average instruction processing rate is very low. An AN/AYK-14, XN-2 configuration is adequate for this node. Two 32K memory modules will be necessary, since a large number of program tasks must be stored. Neither an IOP nor an EAU is considered necessary for this application. A summary of the recommended configuration for Node 40 is given in Table 5-10.

Node 50 — Passive Contact Maintenance, Classification

The requirements for Node 50 are similar to those for Node 40, and the same types of functions are performed. One major periodic function processes passive buoys and requires about 3 seconds of processing time every 15 seconds (worst case). Other operator demand tasks are also performed in this mode. Nodes 40 and 50 represent a processing throughput requirement in excess of that

TABLE 5-10. NODES 40 AND 50 CONFIGURATION

CHASSIS — XN-2		
MODULE	DESIGNATION	QUANTITY
Power Supply	PCM-1	(1)
Memory	32K word	(2)
	or 64K word	(1)
Memory Control	MCM	(1)
General Processor	GPM	(1)
Serial Interface	SIM	(2)
Discrete Interface	DIM	(1)

attainable by a single AN/AYK-14. For this reason and for redundancy considerations, two nodes are necessary. The configuration recommended for Node 50 is identical to that recommended for Node 40 and is summarized in Table 5-10.

#### Node 70 — Navigation and Tactical Processing

Node 70 was chosen as an AN/AYK-14, XN-2 configuration since the processing and memory requirements can easily be met and since this computer is planned for the Integrated Inertial Sensor Assembly/Global Positioning System (IISA/GPS)-based navigation subsystem, which is scheduled to be demonstrated in the BASIC laboratory in the near future. This system will probably have more stringent computational requirements than the S-3A implementation, since it will be used for many platforms. Therefore, considerable excess processing capability over the S-3A requirements is necessary. The AN/AYK-14 will provide this capability. The data update rate between Node 70 and the display subsystem may require a 16-millisecond minor cycle, rather than the current 50 milliseconds. Since the display subsystem as proposed by the Advanced Integrated Display System (AIDS) will have a secondary bus between the integrated control panels and the display processors, it was considered convenient to interface Node 70 to this secondary bus as well as to the system bus. The secondary bus could have the required 16-millisecond minor cycle. This interconnection also provides for the possibility that Node 70 could act as a partial backup in case of Node 81 or 82 failures, and conversely. An XN-2 configuration can fulfill the requirements, with a single 32K-word module and three serial interface module (SIM) channels. At this point, the microprogrammed floating-point capability of the XN-2 appears to be sufficient for the navigation requirements, so an EAU is probably not necessary. An IOP will not be necessary because of the relatively low I/O rate for Node 70. A summary of the recommended configuration appears in Table 5-11.

TABLE 5-11. NODE 70 CONFIGURATION

CHASSIS — XN-2		
MODULE	DESIGNATION	QUANTITY
Power Supply	PCM = 1	(1)
Memory	32K word	(1)
Memory Control	MCM	(1)
General Processor	GPM	(1)
Serial Interface	SIM	(3)
Discrete Interface	DIM	(1)

Nodes 81 and 82 — Display Processing

The configuration for Nodes 81 and 82 is identical and in conformance with the planned configuration of the AIDS program, which is scheduled to be demonstrated by the BASIC laboratory. The display subsystem shown in the system configuration represents the AIDS hardware configuration; therefore, the recommended configuration, summarized in Table 5-12, reflects the computers specified in the preliminary AIDS system specification (Reference 18).

TABLE 5-12. NODES 81 and 82 CONFIGURATION

CHASSIS — XN-1		
MODULE	DESIGNATION	QUANTITY
Power Supply	PCM-2	(1)
Memory	64K word	(1)
	32K word	(1)
Memory Control	MCM	(1)
General Processor	GPM	(1)
Extended Arithmetic	EAU	(1)
Serial Interface	SIM	(2)
Bus Extender	BEM	(1)
Discrete Interface	DIM	(1)



Node 90 — Bus Control/Reconfiguration Monitor

The functions of this node are to monitor all other nodes and to control reconfiguration if another node fails. Node 90 will act as a bus controller in accordance with MIL-STD-1553, and it will act as a backup processor for Nodes 10 and 30. Accordingly, it needs extensive capability for its backup role. The recommended configuration is presented in Table 5-13.

The recommended configurations for each node are summarized in Table 5-14.

5.7.2.2.2 Microcomputer Components. The microcomputers described in the introduction serve primarily as interfaces between subsystem sensors or subsystem processors and the standard serial bus. Several designs for a single-board module for this purpose exist and are being implemented. However, for the BASIC application these devices must have considerable flexibility, since in most cases the devices/subsystems to which they connect will not be the real devices but some kind of simulator. In addition, the interface microcomputers may be required to respond to multiple addresses, as in the case of one

TABLE 5-13. NODE 90 CONFIGURATION

CHASSIS — XN-1		
MODULE	DESIGNATION	QUANTITY
Power Supply	PCM-2	(1)
Memory	32K word	(2)
Memory Control	MCM	(1)
General Processor	GPM	(1)
I/O Processor	IOP	(1)
Serial Interface	SIM	(1)
Discrete Interface	DIM	(1)
Extended Arithmetic	EAU	(1)

TABLE 5-14. NODE HARDWARE CONFIGURATION SUMMARY

AYK-14 MODULES	NODE/QUANTITY									
	10	21/22	30	40	50	70	81/82	90		
CHASSIS	XN-1	XN-1	XN-1	XN-2	XN-2	XN-2	XN-1	XN-1		
POWER SUPPLY				1	1	1				
POWER SUPPLY	(PCM-1)									
POWER SUPPLY	(PCM-2)	1	1	1			1	1		
MEMORY	(32K WDS)	1	3	1	2	1	1	1	2	
MEMORY	(64K WDS)		2*		1*		1			
MEMORY CONTROL	(MCM)	1	1	1	1	1	1	1	1	
GENERAL PROCESSOR	(GPM)	1	1		1	1	1	1	1	
EXTENDED ARITHMETIC UNIT	(EAU)			1			1	1	1	
SERIAL INTERFACE	(SIM)	1	4	1	2	2	2	1	1	
DISCRETE INTERFACE	(DIM)	1	1	1	1	1	1	1	1	
I/O PROCESSOR	(IOP)	1							1	
BUS EXTENDER	(BEM)								1	

\*OPTIONAL IN LIEU OF 32K WORD MODULE(S)

simulator representing multiple sensors, such as in the navigation area. For these reasons, it is recommended that one of the more powerful microcomputer chip sets be used to implement the interface microcomputers for the BASIC laboratory. One such microcomputer interface has already been implemented using the Z-2 chip set. This implementation appears adequate for the projected simulation needs of BASIC and is therefore recommended. A list of Z-2 microcomputer interface/simulator configuration hardware is summarized in Table 5-15. A detailed description of the manner in which 1553 bus data transfers are performed with the Z-2 and interface module (IM) is contained in Reference 17.

5.7.2.2.3 Bus Selection. Two MIL-STD-1553 buses were selected primarily because they both meet the requirements and are designated standard. Furthermore, advanced bus technology and concepts are likely to start with the 1553 as a baseline, thus allowing BASIC to more easily grow into advanced technology bus configurations, such as a higher speed fiber-optic version of 1553.

The primary purposes of the redundant buses (Bus Numbers 2, 3, 4, 5) are to minimize the impact of subsystem configuration changes on the main bus and to provide alternate paths from a computer to a peripheral, so that a computer failure will not result in the isolation of its peripheral from the remainder of the system.

TABLE 5-15. MICROCOMPUTER SIMULATOR CONFIGURATION

MODULE	QUANTITY
Z80-CPU card - 4 MHz	(1)
16K PROM card	(1)
TU-ART I/O interface card	(1) per subsystem
16K RAM card-250 nsec access time	(2) simulated
1553 bus IM card set	(1)
PIO-4 card (for configuration interfacing to status advisory displays only)	(1)

### 5.7.2.3 Control Structure

The objective of this paragraph is to define a general control structure that will efficiently control the timely execution of avionic processes in a multiple computer avionic system. This structure is defined in terms of a number of primitive functions which the control mechanism must perform and a set of parameters, associated with each applications process, on which these primitive functions operate. In concept, the capability to execute the primitive functions must be available within each computer that executes more than one process. Microcomputers that repetitively execute the same process may be considered as functional units and do not require the ability to execute the primitive functions. Most of the implementation structure defined in this paragraph follows the structure recommended in References 4 and 14.

5.7.2.3.1 Control Parameters. The following is a list of control parameters that must be available to each computer for each process that it may execute. These parameters will be stored in a table which is accessed by the control mechanism when each process becomes active.

- a. Task Identifier. A unique process name or identifier
- b. Dependency Tree. If a process generates one or more successor processes the task identifiers of all successor processes and their relative positions in the hierarchical structure make up the dependency tree. Process B is considered a successor of process A in the case in which process B may be executed only if process A has been completed.
- c. Process Authorities. This parameter limits the authority of a process to define for itself and its successor the processor, memory, semaphores, and event responses it will use.
- d. Dispatch Type. Defines whether a process is synchronous, demand, background, or demand/synchronous



- e. Process State. Defines the current state of the process (ready, suspended ready, waiting, suspended waiting, running, or requesting executive service)
- f. Dispatching Priority
- g. Storage Translation Table. All memory in the system is considered as a contiguous virtual memory. This table contains the real addresses of the variables addressed virtually by the process. It also contains other information, such as the protection constraints on the variable and whether the required data segment is local to the computer in which the process is executing or in the memory of a remote computer. Performing this translation is not possible at system generation time, since the real addresses of data may change during the mission as a result of reconfiguration after failure. In addition, once a data word is brought into the local memory of the process, its status changes from remote to local; and future references can be made much more efficiently. The storage translation table is initialized at system generation time and changed by the control mechanism as required.
- h. Processor Identifiers. Identifiers of the physical processors in which the process may run
- i. Semaphore Descriptors. Each resource has a semaphore associated with it which will indicate whether the resource is available or unavailable. This parameter indicates which semaphores are used by the process.
- j. Process Status. Contains state information necessary for the process to run in the computer. This information usually consists of status and working register contents as well as the values of any variables that must be initialized.
- k. Event Response Descriptors. For each event that the process may implicitly or explicitly cause, the event identifier, the allowable processor(s), the event type, and the semaphore descriptors for that event must be listed.
- l. Termination Event Identifiers. Each process causes the identified event as it terminates.
- m. Failure Parameters. This entry contains information indicating which alternate process, if any, is to be executed for each particular class of failures that has been anticipated in the system design.
- n. Interrupt Identifiers. Identifies interrupts honored by the process

**5.7.2.3.2 Control Primitives.** The following is a list of primitive control functions that are assumed to be resident at each computer and available to each process as it executes. These control primitives are executed via calls from the process to the control mechanism, similar to the Executive Service Request of the SDEX-M operating system. Using these primitives, applications programs can be written to implement the control functions outlined in paragraph 3.2.2.4.

- a. Create/Destroy Process. This primitive is used to build or purge a table containing all the parameters listed in paragraph 5.7.2.3.1.
- b. Define/Modify Attribute of Process. This primitive adds or modifies one of the data parameters specific to a process. The data parameters are as defined in paragraph 5.7.2.3.1.
- c. Define Resource. A resource is a processor, a segment of physical storage, a semaphore, or an event. This primitive identifies and bounds a resource needed for execution by the calling process or one of its successor modules.
- d. Relinquish Resource. This primitive releases a previously defined resource so that it may be used by another process.
- e. Define/Modify Attribute of Resource. This primitive defines or modifies the type or quantity of a resource needed by either the process executing the primitive or one of its successor processes.
- f. Define/Modify Sharing of Resource. This indicates which resources may be shared among successor processes identified in the primitive call.
- g. Delegate/Rescind Authority. This primitive enables the process calling the primitive to convey to any successor processes the ability to execute any, some, or all of the previously described "Define" primitives.
- h. Block/Release Process. This is used to stop another process or remove the stop, making it eligible for dispatch.
- i. Semaphore Block/Semaphore Release. Semaphores are used in the normal manner to indicate the availability of resources. The semaphore block/release primitives are used to control access to these resources by processes.
- j. Request Status From. This primitive is used to request messages or status from a process suspended by an event.
- k. Update Status Of. Used to update the status of a suspended process and release the suspension. This primitive and the request status primitive are used to transmit and receive messages between processes.

The primitives described above may be broken into three general categories: process control, which is the management of the suspended, ready, and execution states of processes; definition, by which processes identify resource needs and/or disperse resources from their own pool to their descendant processes; and authority, by which processes convey to descendant processes the ability to execute definition primitives.

The formats for these primitives are shown in Table 5-16.

TABLE 5-16. FORMAT OF PRIMITIVE OPERATORS

<u>Process Primitives</u>			
Block/Release		Process (N)	
Semaphore Block		Process (N)	
Semaphore Release		Process (N)	
<u>Definition Primitives</u>			
Create/Destroy	Process (N)		
Define/Modify	Attribute (N)	of Process (N)	
Define/Relinquish	Resource (N)	of Process (N)	
Define/Modify	Attribute (N)	of Process (N)	of Process (N)
Define/Modify	Sharing	of Resource (N)	between Processes (N) (N)
<u>Authority Primitives</u>			
(Delegate/Rescind Authority for)			
Create/Destroy	of Processes	to/from Process (N)	
Define/Modify	of Attribute	to/from Process (N)	
Define/Relinquish	of Resource	to/from Process (N)	
Define/Modify	of Attribute	of Resource (N)	to/from Process (N)
Define/Relinquish	of Sharing	of Resource (N)	to/from Process (N)



**5.7.2.3.3 Implementation of Control Functions.** Using the control parameters and primitive operators defined in 5.7.2.3.1 and 5.7.2.3.2, configuration-unique application programs can be written for control purposes. For each application process, a kernel executive and a process description table are defined. The kernel executive is a body of common control functions which control the interfaces between hardware and software and between software modules. It has three primary functions: execution of primitives, management of resources (by assigning them to processes and reclaiming them as necessary to support execution time demands), and cooperation with other kernel executives for the purpose of executing primitives requested by remote processes or for performing process or resource reassignment. The process description table is a process control record containing the control parameters described previously; this table is used to describe a process and the process's status, resource needs, and authority to change the process description table of itself or its successor modules.

The kernel executive and its operations using the process description table are the mechanisms by which application programs may affect the state of the multiple computer system. Thus, for each function listed in Table 4-1, a configuration-unique application program can be written which will use the defined primitives within the normal software structure to implement that particular function and communicate with the other functions when necessary.

The control structure described above has two main components for each node. These are the kernel executive, which is the control mechanism itself; and the process description table, which is the data set on which the kernel executive operates. The primitive functions themselves are executed upon request from the application programs.

The above structure is fairly well implemented in the present SDEX/M executive definition, with the exception that SDEX/M controls only a single



computer. However, the necessary multiple-computer primitive functions could be added to the SDEX/M definition, or the present single-computer executive functions could be expanded to include the multiple-computer case. This expansion of the SDEX/M executive into a multiple-computer executive via expansion/addition of multiple-computer primitive functions is recommended for the following reasons:

- a. SDEX/M is already designated the standard executive for the AN/AYK-14.
- b. The structure of SDEX/M can be expanded to cover the structure defined in paragraph 4.4 with only moderate cost.
- c. A full software implementation of the control structure will provide the maximum flexibility and growth potential for the system. This flexibility and growth are obtained at considerable cost in performance, since the most often used control primitives should be implemented in hardware to assure maximum speed. However, the BASIC configuration can "grow into" these hardware implementations if the laboratory is required to simulate a particular platform with well-defined executive requirements. Presently, the flexibility provided by a software implementation outweighs the performance degradation that results from it.

The communication structure can also be implemented in software at moderate cost. The application requirements can be met using software developed from the bus control functions defined by the Digital Avionics Information System (DAIS) program. This software can be viewed as an implementation of the request status and update status primitives defined in paragraph 5.7.2.3.2 of this report. Again, these functions could later be implemented in hardware or firmware if the platform-unique application requirements demand it.

#### 5.7.2.4 Software Configuration

The software for the recommended architecture falls into two general areas: applications software and control (or executive) software. The structure recommended for the control software has been described above, and the application

software structures are represented by the DU's defined in paragraph 3.2.1.2. The objective of this paragraph is to define an implementation of the application software structure.

The DU's previously described are to be implemented in software. However, for the purposes of demonstration in the BASIC laboratory, the DU's need not reflect the actual processing done on the S-3A. They need only reflect the processing time used and the messages either received or transmitted by the process. The majority of processing time is used by periodic processes, and the majority of bus time is used in transmitting periodic messages. Secondary system loading effects occur because of asynchronous messages and demand tasks.

Table 5-2 presents the software allocation structure for the recommended architecture, and the relationships between the software functions and the bus messages are presented in Table 5-17. This table provides general parameters which can be used in the simulation of the recommended architecture via the GCSS simulator. The node and message identifiers in Table 5-17 represent the message structure provided in Table 5-3 of this report. Software functions (DU's) not included in Table 5-17 do not have inputs or outputs.

#### 5.7.2.5 System Operation

Figure 5-7 shows the recommended fault tolerant configuration for a complete BASIC laboratory model. All nodes are AN/AYK-14 configuration XN-1 or XN-2 computers. All buses are implementations of MIL-STD-1553. The display subsystem is configured according to AIDS specifications. Interfaces between subsystems and buses are assumed to be controlled by standard microprocessors (one for each bus). One of the microprocessors would be in standby mode, only operating if a failure occurred in the other. An example of system operation with emphasis on fault detection, isolation, and reconfiguration will be given in the following paragraphs.

TABLE 5-17. DU - MESSAGE RELATIONSHIPS (Sheet 1 of 3)

	SRC DU	SRC NODE	XMT SUB ADR	DST DU	DST NODE	RCV SUB ADR
***						
*	1	10	2	22	82	1
*	1	10	3	22	82	2
*	1	10	6	111	111	1
*	4	10	1	43	70	1
*	5	40	30	22	82	30
*	6	40	1	11	50	1
*	6	40	4	100	100	1
*			5			2
*	6	40	30	34	10	30
*	8	40	7	122	122	1
*	9	40	2	22	82	13
*			3			14
*	9	40	6	119	119	1
*	10	50	1	6	40	1
*	10	50	30	6	40	30
*	10	50	30	22	82	30
*	11	50	2	43	70	3
*			3			4
*	13	40	8	43	70	5
*	13	40	9	22	82	15
*	14	30	1	22	82	6
*	14	30	30	115	115	30
*	15	30	2	22	82	7
*			3			8
*			4			9
*			5			10
*			6			11
*			7			12
*	15	30	30	9	40	30
*	17	30	30	22	82	30
*	17	30	30	43	70	30
*	18	22		105	105	
*			30			30
*	19	22	2	22	82	5
*	19	22	3	105	105	1
*	19	22	30	21	10	30

TABLE 5-17. DU - MESSAGE RELATIONSHIPS (Sheet 2 of 3)

	SRC DU	SRC NODE	XMT SUB ADR	DST DU	DST NODE	RCV SUB ADR
***						
*	22	82	5	24	81	1
*			6			2
*			7			3
*			8			4
*			9			5
*			10			6
*			11			7
*	24	81		101	101	
*	24	81		102	102	
*	24	81		103	103	
*	24	81		104	104	
*	25	82	1	46	10	1
*	25	82	2	19	22	2
*	25	82	3	7	40	3
*	25	82	30	9	40	30
*	25	82	30	10	50	30
*	25	82	30	12	40	30
*	25	82	30	17	30	30
*	25	82	30	18	22	30
*	25	82	30	29	70	30
*	25	82	30	32	21	30
*	25	82	30	45	10	30
*	27	70	1	1	10	3
*	27	70	2	46	10	6
*	27	70	3	32	21	1
*	27	70	5	19	22	1
*	27	70	6	15	30	1
*	27	70	7	17	30	2
*	27	70	8	3	40	2
*	27	70	9	11	50	2
*	27	70	10	22	82	16
*			11			17
*			12			18
*	27	70	30	21	10	30
*	27	70	30	22	82	30
*	30	22	1	32	10	2
*	32	21	1	40	10	1
*	32	21	2	43	70	2



TABLE 5-17. DU - MESSAGE RELATIONSHIPS (Sheet 3 of 3)

	SRC DU	SRC NODE	XMT SUB ADR	DST DU	DST NODE	RCV SUB ADR
***						
*			3			3
*			4			4
*			5			5
*			6			6
*			7			7
*	32	21	30	22	82	30
*	33	21	8	117	117	1
*	35	10	30	116	116	30
*	36	10	30	37	70	30
*	42	82	12	108	108	1
*	42	82	30	43	70	30
*	43	70	4	32	21	23
*	43	70	13	23	82	19
*			14			20
*			15			21
*	43	70	17	42	82	23
*	43	70	30	34	10	30
*	44	70	16	22	82	22
*	45	10	30	22	82	30
*	100	100	9	6	40	4
*			9			5
*	105	105	14	19	22	3
*	106	106	15	28	70	8
*	107	107	16	37	70	9
*	108	108	17	42	82	24
*	110	110	19	28	70	10
*	111	111	20	1	10	5
*	112	112	21	28	70	11
*	113	113	22	28	70	12
*	114	114	23	28	70	13
*	115	115	24	15	30	2
*	117	117	26		21	30
*	123	123	1	25	82	30
*	124	124	1	25	82	30
*	125	125	1	25	82	30
*	126	126	1	25	82	30
						3
---						
?						

5.7.2.5.1 System Initialization. Upon application of power, all of the AN/AYK-14's and microprocessors would automatically initiate a firmware self-check program. This test would culminate in an attempt to communicate its status with the bus control executive program which would be resident in the reconfiguration monitor node. The reconfiguration monitor would be the processor in control of the initialization process. Its self-check process complete, this processor would then attempt to load its memory from the computer control unit (CCU) or loader/verifier units. If this is successful, the reconfiguration monitor node would then assume control of the main system bus and begin execution of the bus control process. If the various units are operational, the reconfiguration monitor will then supervise the loading of the various computers from the bulk store unit, if necessary. (The microcomputers may have firmware programs.) Initially, diagnostic programs will be loaded and executed, with results sent back to the reconfiguration monitor. If all diagnostic tests are positive, the operational programs will be loaded; and execution will begin.

5.7.2.5.2 On-line Detection. Continuous built-in test equipment (BITE) hardware in the AN/AYK-14 will monitor the functions listed in Table 5-18. On-line detection is not presently implemented for any commercially available microcomputer.

5.7.2.5.3 Scheduled Test. The AN/AYK-14 in-flight performance monitoring (IFPM) program responds to an interrupt periodically generated by the hardware built-in-test (BIT) timer and is specified to detect 98 percent of all faults in the system. In addition to the IFPM modules, generation of a periodic message to the remainder of the system indicating the status of the computer will be necessary. If this message is not properly transmitted and received by the rest of the system, assume that the computer has failed.

The microcomputers in the system should also perform a periodic self-test function and generate status messages to the system. Where two redundant

TABLE 5-18. AN/AYK-14 HARDWARE BITE

- Memory Parity
- Memory Protect
- Memory Channel Timeout
- Power Monitoring
- Overtemperature Monitoring
- Bus Timeouts
- I/O Channel Parity
- I/O Channel Timeout
- Manchester Code Format Verification
- BIT Timer
- BIT Indicator

microcomputers perform the same interface function (for different buses), their self-test programs and status messages should be identical so they can be compared by one or more receiving monitors and so faults can be more easily detected and isolated.

5.7.2.5.4 Redundancy Testing. The primary redundant resources in the system are the reconfiguration monitor node, the redundant interface microcomputers, the redundant 1553 buses, and extra memory modules in all the AN/AYK-14 computers. The reconfiguration monitor is testing itself periodically and generating a status message which is to be sent to all other nodes. The redundant interface microcomputers are conducting self-test programs periodically and sending status messages over their respective buses to either the reconfiguration monitor or to the computer controlling the secondary bus. The bus controller receives both of the status messages and compares them to determine the status of the microcomputers. The redundant buses are tested every time a message is transferred over them. If repeated errors occur within messages transmitted from different sources, assume that the bus is faulty.

#### 5.7.2.5.5 Isolation

##### CPU/IOP Failures

If a CPU or an IOP of an AN/AYK-14 or the CPU of a microcomputer fails, then the entire unit must be isolated from the remainder of the system. If the failure has not occurred in a unit which is a bus controller, isolation can be accomplished by the appropriate bus controller program; the bus controller program will merely refuse to read from or write to the faulty unit. If the failure is such that the bus is not usable, that bus may be considered as failed and may switch to its backup bus. Alternatively, removal of power from the faulty unit may be possible to determine if the bus is usable. If the Advanced Aircraft Electrical System (AAES) is providing electrical power, power removal can be effected by an appropriate request to the AAES control computer. Otherwise, special-purpose hardware must be employed.

If a CPU fails in one of the units that is a secondary bus controller, the reconfiguration monitor will be informed of this via status messages of the failed units and any AN/AYK-14 units that connect to the same secondary bus. The reconfiguration monitor will then prepare the backup computer to assume the duties of bus controller for the secondary bus; shut down the faulty computer by means of a signal on a discrete wire; and signal the backup computer, via the bus, to imititate bus control functions.

If a failure occurs in the reconfiguration monitor (which is the bus controller for the main system bus), this situation will be made known to the rest of the system via the status messages (or lack of them) from the reconfiguration monitor. Discretes from three of the other AN/AYK-14 computers will activate a majority voting circuit which will shut down the reconfiguration monitor when any two of the three discrete signals are enabled. One of the other AN/AYK-14's will then begin executing the prestored bus controller program.



### Memory Failures

A memory failure in an AN/AYK-14 will be detected by BIT and/or the IFPM and will be signaled by an interrupt. In any case, the faulty module will be isolated by an interrupt-service routine which will prevent access to that module until reconfiguration recovery can be effected. If the memory of a microcomputer fails, the entire microcomputer will be isolated.

### I/O Channel Failures

The interface microcomputers will make reasonableness checks on their sensor subsystem input as often as is feasible. If the input appears in error, the microcomputers will notify the bus controller by means of a status message. The bus controller will compare this condition with that of the microcomputer on the other bus. If both messages agree that the sensor data are in error, the bus controller will not allow any more data transmission from that sensor.

#### 5.7.2.6 Recovery

5.7.2.6.1 CPU/IOP Failures. If an entire computational unit has been isolated from its interfaces, then the operational processes previously performed by that unit must be executed elsewhere. The unit that executes these added processes must have the following characteristics:

- a. The unit must have access to the peripherals/interfaces of the failed machine.
- b. It must have access to the program code which performs the same function as the processes executed by the failed machine.
- c. It must have memory sufficient to store the extra code.
- d. It must have available time to execute the extra code.

If one of the microcomputers has failed, the redundant microcomputer will replace it. The redundant microcomputer has the same interfaces as its failed counterpart. The appropriate code has been prestored in its memory, and it

will execute this program as its highest priority, thus ensuring that sufficient time will be available. The self-test programs will continue to execute, but at a lower iteration rate. If Nodes 10, 21 and 22, 30, or 50 fail, they will be replaced by the reconfiguration monitor. The reconfiguration monitor will, upon indication that the node has been successfully isolated, load itself with the appropriate programs from the bulk store. All required data will be marked "nonresident" in the data translation tables, and the programs will start execution at the beginning of the next major cycle. Self-test programs and bus control will continue on the reconfiguration monitor, but the iteration rate will be lower.

If Nodes 40, 70, 81, or 82 fail, they will be replaced by Nodes 50, 21, 82, and 81, respectively, so that access to the proper interfaces will be maintained. If Nodes 40 and 70 do not have the capability to execute the extra processes as well as their own operational processes, it may be necessary to perform a dual reconfiguration using the reconfiguration monitor. For example, if Node 70 fails, it must be replaced by Node 21 or 22, since only these nodes have the proper interfaces. However, if Node 21 cannot perform both sets of processes, then the reconfiguration monitor must be used to replace Node 21. Nodes 50, 21, and 22 have no operational interface with their respective secondary buses and are only connected to those secondary buses for purposes of reconfiguration.

The reconfiguration monitor performs no operational function other than bus control. Therefore, bus control is the only function transferred if the reconfiguration monitor fails.

The bulk store again performs no operational function and so requires no reconfiguration process. However, implementation of the bulk store in a number of small modules may be more desirable to reduce the probability of a total failure.

In general, the replacement process proceeds as follows: an interrupt will be generated in the replacement computer; the required programs will be read

into the replacement unit from the bulk store; the control parameters (iteration rates, priorities, etc.) will be given to the executive program from the same bulk store — these are predefined for each type of failure; all data required by the program will be marked "nonresident"; system variables will be updated by the global executive to reflect the new configuration; the interrupted program will be gracefully aborted; and the program with the highest local dispatch priority will begin execution in the replacement computer.

5.7.2.6.2 Memory Failures. If a memory fails in a microcomputer, it should be treated as a CPU failure as described above. If the memory module in an AN/AYK-14 fails, it will be replaced by a resident spare memory module. If the failed segment contained program code or constants, these should be replaced from data in the bulk store.

Variable data will be lost and must be regenerated. For this reason, storing important and slowly changing data, such as system track data, in two computers may be desirable to assure that these data can be recovered quickly in the case of a memory failure.

5.7.2.6.3 I/O Failures. The primary mechanism for recovery from I/O failures is the automatic switchover to the redundant bus, as provided for by the 1553 bus structure and bus control software.

#### 5.7.2.7 Conclusions

This section has presented a recommended system configuration for BASIC and the rationale for developing this configuration. First, the decompositional units derived from the S-3A functional requirements were grouped into related functional areas and assigned to processing nodes. The data flow among the nodes was extensively analyzed, and a set of 1553 messages was defined to accommodate this data flow. The data rate resulting from this message structure was calculated and found to be moderate, allowing a wide growth margin

before new buses must be added. The processing requirements for each node were analyzed and a hardware configuration for the node developed. Software execution parameters were defined and presented in a format suitable for input to a hardware-on-software simulation effort.

Finally, an example of overall system operation was presented, with emphasis on fault tolerance and reconfiguration.



SECTION 6

IMPLEMENTATION SUBSET

6.1 OBJECTIVE

The objective of this section is to define a subset of the recommended architecture which may be implemented in a short time with reasonable cost and which will demonstrate and verify the major concepts embodied in the full architecture.

The subset defined in this section will demonstrate both flexibility and fault tolerance and will provide an indication of the capability of the full architecture in the following areas:

- Computational power
- Operator interface
- Control mechanism efficiency
- Communication mechanism efficiency

In addition to the demonstration aspects, the configuration recommended will be available to provide system integration services to both subsystem technology developers and system platform users.

Funding and time constraints dictate that the configuration recommended contain only available hardware and that software generation be held to a minimum.

6.2 APPROACH

In order to make the system as generic and usable as possible, the subset of the architecture shown in Figure 6-1 was chosen. This configuration contains modes used for any avionic application: communications (Node 21), navigation (Node 70), display and control (Nodes 81 and 82), and the bus controller and re-configuration monitor (Node 90). In addition, the MAD mode (Node 30) was

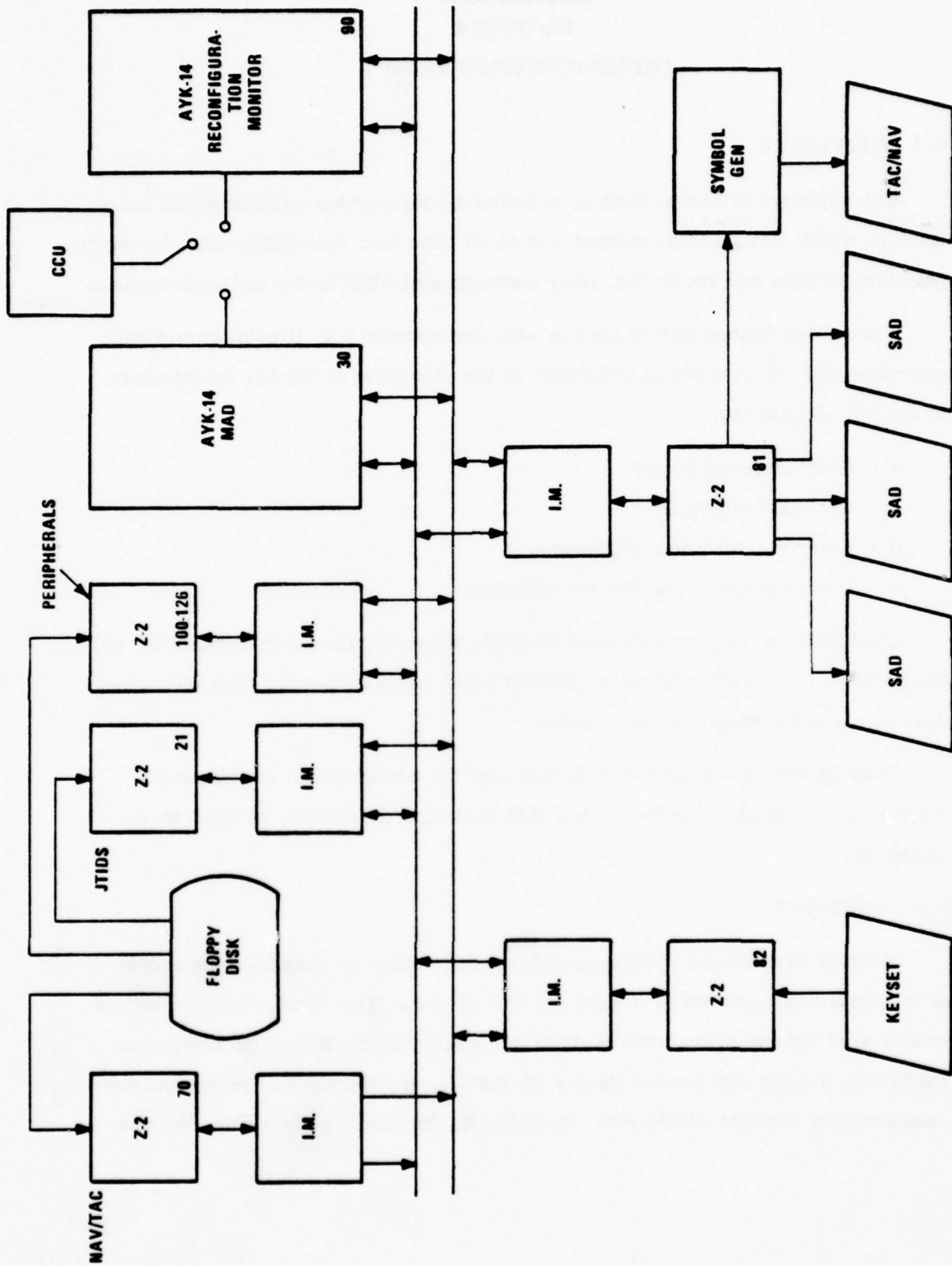


Figure 6-1. Implementation Subset

chosen because an existing MAD program, written in SPL/I, exists and can be used without major modification. In addition, Node 30 provides convenient growth capability and may act as a backup for Node 90 without compromising its operational capability. The MAD subsystem also does not require sophisticated, high-speed signal processing.

The flexibility of the system may be demonstrated through the choice of those subsystems which the nodes are to represent. For example, the navigation subsystem on the S-3A does not contain IISA or ASW support aircraft carrier (CVS), yet Node 70 simulates these systems at the interface without modification of the message structure or data rates derived for the full, S-3A-based architecture. Similarly, the bus interface defined for Node 21 in the full architecture will accommodate the postulated Joint Tactical Integrated Data System (JTIDS) interface without modification. The AIDS subsystem is also simulated at the interface by Nodes 81 and 82. Thus, the node partitioning and message structure derived from the S-3A-based requirements are flexible enough to accommodate three advanced technology subsystems without major modification.

A limited amount of fault tolerance making maximum use of existing IFPM and bus error processing routines is incorporated into the system. Provisions are made for periodic system self-check and subsequent load of reconfiguration programs, although these programs are not defined in detail. It is anticipated that these programs will start out very simple and be made more sophisticated as experience is gained with the system. Bus traffic and conflicts can be measured and compared against those predicted by analysis and simulation in order to aid in the evaluation of the architecture.

In terms of service to users, several new technology subsystems (AIDS, TIES, IISA) can be simulated in more detail and eventually incorporated into the recommended system.

After selection of a configuration, a simple scenario was chosen as a demonstration problem. The scenario was abstracted from the set of operational sequence diagrams (Appendix B) developed for use by the V/STOL program. These operational sequence diagrams depict the interactions between the operators and the avionic subsystems for a V/STOL Type A aircraft ASW mission. The abstracted sample problem is considerably simplified (therefore, unrealistic) because of hardware and manpower constraints, but it may serve as a starting point for a more realistic demonstration if desired.

In order for the scenario to be useful, it must include human interaction. Therefore, the scenario cannot be entirely predefined but must include asynchronous responses to operator Keypad inputs. The next step in the demonstration, then, was the definition of the allowable operator inputs and the responses to them. This, together with the requirements for preprogrammed synchronous message generation, led to the definition of the specific functions that must be performed by the modes in the scenario, and the specific contents of the messages passed over the bus.

Finally, general specifications were developed for the software required to implement the functions. In the specification of the software, available high-level language (HLL) software was specified where available, or where nearly available (that is, SDEX-M, Bus Controller Software, AN/AYK-14 IFPM programs). Assume that software will be developed in HLL using structured programming techniques wherever feasible.

### 6.3 SUBSET CONFIGURATION

As previously stated, Figure 6-1 shows the selected subset configuration. In general, the nodes have four major functions: they perform executive functions which control the scheduling and dispatching of required programs; they execute synchronous and asynchronous modules which are necessary to the execution of the chosen scenario; they perform I/O operations (which consist of the



generation and reception of those messages assigned to them in Table 5-3) and generation and reception of asynchronous commands which occur as a result of operator action; finally, they perform test and recovery functions. The assumption is that the executive functions of the AN/AYK-14's (Nodes 30 and 90) will be implemented using the SDEX-14 or SDEX-M executive. The Z-2's (Nodes 70, 21, 100 to 126, 81, and 82) will require that executive software be generated).

The major synchronous processes necessary for the scenario are as follows:

- Simulation of aircraft, fix, and track position updates by Node 70
- Display processing by Nodes 81 and 82
- MAD processing by Node 30
- Bus control by Node 90
- Peripheral I/O simulation by Nodes 100 to 126
- JTIDS I/O simulation by Node 21

The synchronous I/O messages are as defined in Table 5-3. In general, the data content is meaningless and is only required to determine bus utilization. The data content is defined only for those messages necessary for the enactment of the scenario.

All the nodes execute self-test programs every minor cycle and format the results into a self-test message sent to Node 90. (Node 90 sends a self-test message to Node 30.) If a failure is sensed, reconfiguration is attempted. Reconfiguration is accomplished by isolating the failed node from the system loading reconfiguration programs from the mass memory into Node 90 and reinitializing Node 90. If Node 90 fails, it is isolated from the system, and Node 30 takes over the bus control function. This bus control program must be resident in Node 30 at all times so that it may communicate with the mass memory, if possible.

## 6.4 SCENARIO

### 6.4.1 Objective

The objective of this section is to define a highly simplified, flexible scenario which will demonstrate and verify the performance of the implementation subset. The scenario will be abstracted from the operational sequence diagrams developed for the V/STOL Type A ASW mission (Appendix B), but the hardware and time constraints prohibit the complete enaction of any significant portion of Appendix A.<sup>1</sup>

### 6.4.2 Description

The recommended scenario involves only the TACCO. He may simulate the placement of up to 32 buoys and their type, either LOFAR or DIFAR. If the LOFAR buoys are within a certain range of a simulated submarine, a contact alert will result. If a DIFAR buoy is within a certain range, a contact alert plus a bearing will be displayed. Convergence zones are ignored. Fixes may be entered and a track will be displayed based on the last three fixes on a single contact. The objective of the scenario is to obtain a MAD contact, which will occur if the MAD processing is activated and the aircraft signal passes over the submarine within a predefined radius. There may be more than one target in the system which may represent the same target sensed by different subsystems at different times.

## 6.5 SYSTEM DATA

### 6.5.1 Objective

The objective of this paragraph is to present a general description of the major data structures that are necessary for enaction of the selected scenario. Both static data, which do not change over the course of the scenario, and dynamic data are included.

---

<sup>1</sup> Appendix A is not contained in this volume but is available from NAVAIRDEVCON Computer Systems Technology Division, Code 502.

### 6.5.2 Static Data

Static data will normally not change during a scenario enaction, unless reconfiguration occurs because of failure. The static data are preprogrammed and loaded at initialization. If reconfiguration does occur, a different set of static data are loaded when specific nodes are reinitialized to perform additional functions. There are four major categories of static data: program code, some message buffers, initialization control data for the executive, and the message control data base (PALEFAC) as described in DAIS Mission Software Product Specification Executive, Vol. 2, Bus Control SA201302, of 1 November 1977.

#### 6.5.2.1 Program Code

Both normal and reconfiguration programs must be coded and resident in the system. Normal programs are loaded at the time of or prior to system initialization. Reconfiguration programs are loaded from the disk, through Node 21, and over the MIL-STD-1553 bus as required. A set of reconfiguration programs is postulated to exist for each of the various classes of failures that are considered for reconfiguration. Initially, the total failure and isolation from the bus of Node 70, Node 21, Node 30, Node 90, and Node 82 are the only failure classes considered. Thus, there are five program sets required; these may overlap, as is anticipated in the case of the AN/AYK-14's (Nodes 90 and 30). That is, most of the normal program set for Node 30 is used as the reconfiguration set which Node 90 will execute in its function of Node 30 backup.

#### 6.5.2.2 Static Message Buffers

Some of the messages defined in Table 5-3 are not used in the scenario action; thus, there is no need to either change them or make their contents meaningful. Therefore, the buffers should be loaded once at initialization time and thereafter transmitted periodically. Some predefined pattern should be loaded so that correct operation may be checked if desired.

#### 6.5.2.3 Initialization Control Data

These data are available to the local executives to control initialization (or reinitialization) of the nodes. These data include such things as program load locations, task identifiers, dispatching type and priority, and semaphore linkages.

#### 6.5.2.4 PALEFAC Data Base

These data are used to control the message structure over the MIL-STD-1553 bus and are described in the DAIS Bus Control Specification.

#### 6.5.3 Dynamic Data

The scenario requires periodic position updates and display of three types of objects: contacts, sonobuoys, and the aircraft itself. In order to enact the scenario, the operator must enter both commands and data via the Keyset, and the displays must show both symbols and alphanumerics. To accommodate these requirements, four dynamic file structures are defined: the track/contact file, the fix/contact file, the sonobuoy file, and the text file.

##### 6.5.3.1 Track/Contact File

The track/contact file contains the position, speed, heading, and so forth, of the aircraft and any simulated targets. This file is updated periodically according to the following rules:

- If one or more fly-to points (FTP's) has been entered, the aircraft will proceed to the first entered FTP at a preset speed and altitude. If the FTP is reached, the aircraft proceeds to the next entered FTP (if one exists) or flies a circular pattern at a preset speed, altitude, and radius about the last entered FTP.
- All targets proceed at a preselected heading at constant speed. If the targets exit the display, they start again at a preselected point. The format for track entries is shown in Figure 6-2.

##### 6.5.3.2 Fix/Contact File

Fixes are entered as a result of an enter fix command. Fix entries have a position indicated by the hook positions on the display. If the fix position is within



TRACK FILE (Contains current position of all targets)

- Track ID
- Fix 1 ID
- Fix 2 ID
- Fix 3 ID
- Target Latitude (2 words)
- Target Longitude (2 words)
- Target Grid Coordinates (U) (2 words)
- Target Grid Coordinates (V) (2 words)
- Bearing
- Target Velocity
- Target Classification
- Display Cue

FIX FILE

- Contact ID
- Fix ID
- Fix Latitude (2 words)
- Fix Longitude (2 words)
- Fix Grid Coordinates (U) (2 words)
- Fix Grid Coordinates (V) (2 words)
- Classification
- Display Cue
- Time of Fix (2 words)

SONOBUOY FILE

- Sonobuoy ID
- Grid Coordinates (U) (2 words)
- Grid Coordinates (V) (2 words)
- Vector Endpoint (U) (2 words)
- Vector Endpoint (V) (2 words)
- Time of Update (2 words)
- Display Cue
- ?

Figure 6-2. Data Files

a preset radius of a track in the track file, the fix ID is entered into the appropriate position in the track file entry. Three fixes entered into the track file will cause a track symbol to be displayed. The format for a fix entry is shown in Figure 6-2.

#### 6.5.3.3 Sonobuoy File

The sonobuoy file contains the positions of sonobuoys inserted into the system by means of the drop sonobuoy command. Sonobuoy position is determined by the aircraft position at the time the command is executed. The format for a sonobuoy file is shown in Figure 6-2.

#### 6.5.3.4 Text File

This file contains text to be displayed at any or all of the displays and is merely a file of ASCII character codes, two per 16-bit word.

The operations performed on these files are described in more detail in paragraph 6.7.

### 6.6 FUNCTIONAL DESCRIPTION OF NODES

#### 6.6.1 Node 70

##### 6.6.1.1 General Description

Node 70 simulates the navigation and tracking subsystem of the full architecture. This node calculates aircraft position heading, pitch, roll, and steering information. It also keeps track of sonobuoy positions and contact and fix information. It transmits this information to almost all other nodes in several groups of periodic messages and responds appropriately to asynchronous messages (commands) originating from the Keyset.

In the scenario, Node 70 simulates the position and motion of the aircraft and targets, updates the fix and sonobuoy positions, determines the flight path of the aircraft, determines if a contact has been made from the relative position of the simulated targets and sonobuoys, and keeps time.

The hardware consists of an Interface Module (IM), a Z-2 microcomputer, and an interface with the floppy disk. Use of this disk will probably not be required for anything other than initial program load.

#### 6.6.1.2 I/O Description

The input and output messages making up the Node 70 interface in the implementation subset are shown in Tables 6-1 and 6-2, respectively. Asynchronous message formats are dependent upon the particular command they represent and are described in Tables 6-3 and 6-4. Formats and data content for Node 70 synchronous input and output messages are shown in Tables 6-5 and 6-6, respectively.

TABLE 6-1. NODE 70 INPUT MESSAGES

	SRC NODE	RCV SUB ADR	WORDS PER SECOND	PERIOD IN MSEC	CYC CODE	WORD COUNT
***						
*	21	2	3000	2000	0	32
*		3			0	32
*		4			0	32
*		5			0	32
*		6			1	32
*		7			2	16
*	30	30	3	0		3
*	82	30	14	0	4	14
*	82	30	25	0		5
*	90	30		0		3
*	106	8	240	50	0	12
*	107	9	120	200	2	24
*	110	10	10	200	2	2
*	112	11	20	200	2	4
*	113	12	60	200	2	12
*	114	13	160	50	0	8
						1

TABLE 6-2. NODE 70 OUTPUT MESSAGES

	DEST. NODE	XMT SUB ADR	WORDS PER SECOND	PERIOD IN MSEC	CYC CODE	WORD COUNT
***						
*	21	3	210	100	1	21
*	21	4	960	1000	0	32
*	30	6	300	50	0	15
*	30	7	420	50	0	21
*	82	10	1240	50	0	19
*		11			0	21
*		12			0	21
*	82	13	1500	2000	0	30
*		14			0	30
*		15			0	30
*	82	16	320	1000	1	32
*	82	17	100	2000	2	11
*	82	30	16	0		16
*	90	29	160	50	0	8
						1

#### 6.6.1.3 Software Functional Description

The following software module descriptions are intended to describe the structure and interaction of the software in a general but comprehensive manner. These descriptions are not intended to be specifications but rather a general outline of what the system is required to do.

6.6.1.3.1 Control Modules. Since the Z-2 configurations lack a preprogrammed executive, they will require an operating system of sorts. This executive will be required to initialize the system (including the load of relevant programs and data from the floppy disk), to handle interrupts, to read and write from the bus, and to schedule and dispatch modules. Figures 6-3 through 6-8 describe the executive functions implemented by software.



NADC-79161-40

TABLE 6-3. NODE 70 COMMAND INPUTS  
(RCV SUB ADR 30)

COMMAND	WORD NO.	FORMAT	CONTENT
Initialize NAV	0	Bit String	Command ID
Terminate NAV	0	Bit String	Command ID
Mark Time	0	Bit String	Command ID
Read Latitude/Longitude	0	Bit String	Command ID
	1	Integer	X Hook Position
	2	Integer	Y Hook Position
Enter Fix	0	Bit String	Command ID
	1	Integer	X Hook Position
	2	Integer	Y Hook Position
Enter FTP	0	Bit String	Command ID
	1	Integer	X Hook Position
	2	Integer	Y Hook Position
Recenter	0	Bit String	Command ID
Acknowledge Contact	0	Bit String	Command ID
	1	Integer	X Hook Position
	2	Integer	Y Hook Position
New Minor Cycle	0	Bit String	Command ID
Controller Status	0	Bit String	Command ID
Destroy Fix	0	Bit String	Command ID
	1	Integer	X Hook Position
	2	Integer	Y Hook Position
Destroy FTP	0	Bit String	Command ID
	1	Integer	X Hook Position
	2	Integer	Y Hook Position

TABLE 6-4. NODE 70 ASYNCHRONOUS OUTPUTS  
(XMT SUB ADR 30)

COMMAND	WORD NO.	FORMAT TYPE	CONTENT
Contact Alert	0	Bit String	Identifier
	1	Bit String	Fix 1 ID
	2	Bit String	Fix 2 ID
	3	Bit String	Fix 3 ID
	4,5	Floating Point	Target Latitude, deg.
	6,7	Floating Point	Target Longitude, deg.
	8,9	Floating Point	U Target Grid Coordinates, nm
	10,11	Floating Point	V Target Grid Coordinates, nm
	12	Scaled	Target Bearing, deg.
	13	Scaled	Target Velocity, knots
	14	Bit String	Target Classification
	15	Bit String	Target Display Cue

6.6.1.3.2 Operational Modules. These modules deal with the operational aspects of the scenario. They perform position calculations, format outputs, manage the fix and track files, respond to commands, and write messages to the bus. Figures 6-9 through 6-29 describe these operational functions. These modules operate primarily on three data files: the track file, the fix file, and the sonobuoy files. The contents of a typical entry for each file are shown in Figure 6-2. The number of entries in each file is contained in the variables NUM\_TRACKS, NUM\_FIXES, and NUM\_SONOBUOYS.

6.6.1.3.3 Test and Reconfiguration Modules. Node 70 executes a self-test program every minor cycle and reports the result to Node 90 by means of a self-test output message, as described in Table 6-7. Node 70 also receives the Node 90 self-test output and returns it to Node 30 on request. This occurs if Node 30 has received a self-test message from Node 90 indicating a Node 90 failure and is for verification. Figures 6-24 and 6-25 describe the test and recovery function.

TABLE 6-5. NODE 70 INPUT MESSAGE FORMATS

SOURCE NODE	XMT SUB ADR	RCV SUB ADR	TYPE	FORMAT TYPE	WORD NO.	CONTENT
21 (JTIDS)	2	2	Static	Integer	1	Message ID
				Floating Point	2, 3	Aircraft Latitude
				Floating Point	4, 5	Aircraft Longitude
				Floating Point	6, 7	CMD Altitude
				Floating Point	8, 9	Grid Azimuth
				Floating Point	10, 11	CMD HDG
				Floating Point	12, 13	CMD Course
				Floating Point	14, 15	TACAN Range
				Floating Point	16, 17	TACAN BRG
				Floating Point	18, 19	Time of Day
				Bit String	20	TACAN JTIDS Mode/Status
				Floating Point	21, 22	Runway HDG
				Floating Point	23, 24	Glide Slope Error
				Floating Point	25, 26	Lateral (Localizer) Error
				Bit String	27	Landing Alerts
				Floating Point	28, 29	Ref. Latitude
				Floating Point	30, 31	Ref. Longitude
				Bit String	32	Waypoint ID
	3-7	3-7	Static	Bit String	1, 17	Track ID
				Bit String	2, 18	Fix 1 ID
				Bit String	3, 19	Fix 2 ID
				Bit String	4, 20	Fix 3 ID
				Floating Point	5, 6, 21, 22	Latitude
				Floating Point	7, 8, 23, 24	Longitude
				Floating Point	9, 10, 25, 26	Grid Coordinate (U)
				Floating Point	11, 12, 27, 28	Grid Coordinate (V)
				Scaled	13, 29	Bearing
				Scaled	14, 30	Velocity
				Bit String	15, 31	Classification
				Bit String	16, 32	Display Cue
106	1	8	Static	Bit String	1-12	Preset Pattern
107	1	9	Static	Bit String	1-24	Preset Pattern
110	1	10	Static	Bit String	1-2	Preset Pattern
112	1	11	Static	Bit String	1-4	Preset Pattern
113	1	12	Static	Bit String	1-12	Preset Pattern
114	1	13	Static	Bit String	1-8	Preset Pattern

TABLE 6-6. NODE 70 OUTPUT MESSAGE FORMATS

DEST. NODE (S)	XMT SUB ADR	RCV SUB ADR	TYPE	FORMAT TYPE	WORD #	CONTENTS
21 30 82	3 7 11	1 2 17	DYNAMIC	BIT STRING FLOATING PT. ↓ INTEGER FLOATING PT FLOATING PT INTEGER SCALED INTEGER ↓	0 1,2 3,4 5,6 7,8 9 10,11 12,13 14 15 16 17 18 19 20	GROUP ID LATITUDE OF A/C (DEG) LONGITUDE OF A/C (DEG) GRID CO-ORD (U) - MILES GRID CO-ORD (V) - MILES BARO ALTITUDE FT. RADAR ALT - FT COMMAND ALT - FT COMMAND ALT ERROR - FT GROUND SPEED - FT/SEC TRUE AIR SPEED - FT/SEC INDICATED AIR SPEED - FT/SEC COMMAND AIR SPEED - FT/SEC COMMAND AIR SPEED ERROR - FT/SEC MACH NO.
21 82	4 16	23 22	DYNAMIC	BIT STRING ↓ FLOATING POINT ↓ SCALED SCALED BIT STRING BIT STRING	0,16 1,17 2,18 3,19 4,5;20,21 6,7;22,23 8,9;24,25 10,11;26,27 12,28 13,29 14,30 15,31	TRACK ID FIX 1 ID FIX 2 ID FIX 3 ID TGT LATITUDE DEG TGT LONGITUDE DEG TGT GRID CO-ORD (U) MILES TGT GRID CO-ORD (V) MILES TGT BEARING DEG TGT VELOCITY KNOTS TGT CLASSIFICATION DISPLAY CUE
21 30 82	3 7 11	1 2 17	DYNAMIC	BIT STRING FLOATING PT. ↓ INTEGER FLOATING PT FLOATING PT INTEGER SCALED INTEGER ↓	0 1,2 3,4 5,6 7,8 9 10,11 12,13 14 15 16 17 18 19 20	GROUP ID LATITUDE OF A/C (DEG) LONGITUDE OF A/C (DEG) GRID CO-ORD (U) - MILES GRID CO-ORD (V) - MILES BARO ALTITUDE FT RADAR ALT - FT COMMAND ALT - FT COMMAND ALT ERROR - FT GROUND SPEED - FT/SEC TRUE AIR SPEED - FT/SEC INDICATED AIR SPEED - FT/SEC COMMAND AIR SPEED - FT/SEC COMMAND AIR SPEED ERROR - FT/SEC MACH NO.
21 82	4 16	23 22	DYNAMIC	BIT STRING ↓ FLOATING POINT ↓ SCALED SCALED BIT STRING BIT STRING	0,16 1,17 2,18 3,19 4,5;20,21 6,7;22,23 8,9;24,25 10,11;26,27 12,28 13,29 14,30 15,31	TRACK ID FIX 1 ID FIX 2 ID FIX 3 ID TGT LATITUDE DEG TGT LONGITUDE DEG TGT GRID CO-ORD (U) MILES TGT GRID CO-ORD (V) MILES TGT BEARING DEG TGT VELOCITY KNOTS TGT CLASSIFICATION DISPLAY CUE



TABLE 6-6. NODE 70 OUTPUT MESSAGE FORMATS (Continued)

DEST NODE(S)	XMT SUB ADR	RCV SUB ADR	TYPE	FORMAT TYPE	WORD #	CONTENTS
82	12	18	DYNAMIC	BIT STRING SCALED ↓	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14, 15 16 17, 18 19, 20	GROUP ID DEG MAG HDG DEG GRID AZIMUTH DEG CMD HDG DEG CMD HDG ERR. DEG CMD COURSE DEG CMD COURSE ERR DEG WIND SPD DEG WIND BEARING DEG ANGLE OF ATTACK DEG SLIP INDICATOR DEG BEARING TO DEST DEG RANGE TO DEST N.M. TACAN RANGE N.M. TACAN BEARING DEG TACAN DEVIATION DEG TIME OF DAY HRS TIME TO GO HRS
82	13 14 15	119 20 21	DYNAMIC	BIT STRING ↓ FLOATING POINT ↓ BIT STRING ↓	0 : 15 1 : 16 2 : 17 3, 4 : 18, 19 5, 6 : 20, 21 7, 8 : 22, 23 9, 10 : 24, 25 11 : 26 12 : 27 13, 14 : 28, 29	GROUP ID CONTACT ID FIX ID TGT LATITUDE DEG TGT LONGITUDE DEG TGT GRID CO-ORD (U) N.M. TGT GRID CO-ORD (V) N.M. CLASSIFICATION DISPLAY CUE TIME OF FIX
82	17	23	DYNAMIC	BIT STRING FLOATING PT FLOATING PT FLOATING PT	0 1, 2, 3, 4 5, 6, 7, 8 9, 10	GROUP ID (SONOBUOY ID) SONOBUOY GRID CO-ORD (U,V) N.M. VECTOR END POINTS (U,V) N.M. DISPLAY CUE N.M.
90	29		5	BIT STRING ↓	0 1 2 3 4 5 6 7	NODE ID, MAJOR CYC. =, MINOR CYCLE = STATUS WORD 1 STATUS WORD 2 SYSTEM CONFIGURATION ID SELF TEST RESULTS WORD OLD SEQUENCE WORD (LAST WORD 7 SENT) TBD NEW SEQUENCE WORD

Figure	Function	Input	Output	Notes
6-3	Executive Function 1	Input 1	Output 1	...
6-4	Executive Function 2	Input 2	Output 2	...
6-5	Executive Function 3	Input 3	Output 3	...
6-6	Executive Function 4	Input 4	Output 4	...
6-7	Executive Function 5	Input 5	Output 5	...
6-8	Executive Function 6	Input 6	Output 6	...

Figures 6-3 through 6-8. Node 70 Executive Functions

NADC-79161-40  
MODULE NAME: INITIALIZE NAV SYSTEM

DATA REQUIRED

NONE

DATA PRODUCED

NONE

INITIALIZE NAV SYSTEM

THIS PROCESS IS ACTIVATED BY A POWER ON CONDITION (IF POSSIBLE). IT LOADS PROGRAMS AND PRESET DATA FROM THE FLOPPY DISK AND SETS AND NECESSARY INITIAL VALUES. IT ALSO INITIALIZES THE INTERRUPT STRUCTURE AND SCHEDULES THE "OUTPUT PROCESSING" AND "SELF TEST" PERIODIC MODULES. FINALLY, IT CALLS THE "DISPATCH" MODULE TO INITIATE PROCESSING.

END

Figure 6-3

MODULE NAME: INTERRUPT HANDLER

DATA REQUIRED

TYPE OF EVENT (EXTERNAL OR BUS INPUT)

DATA PRODUCED

NONE

INTERRUPT HANDLER

THIS FUNCTION SAVES THE STATE OF THE MACHINE AND TRANSFERS CONTROL TO THE PROPER INTERRUPT RESPONSE MODULE

LOCK OUT ALL FURTHER INTERRUPTS  
SAVE PROGRAM CTR  
SAVE STACK POINTERS  
SAVE REGISTER CONTENTS  
CASE INTERRUPT TYPE (EXTERNAL OR BUS INPUT)  
  . "EXTERNAL INTERRUPT"  
  . "READ"  
ENDCASE  
RESTORE MACHINE STATE  
ENABLE INTERRUPTS  
TRANSFER TO PROGRAM CTR

END

Figure 6-4

NADC -79161-40  
MODULE NAME: DISPATCH

DATA REQUIRED

INIT\_FLAG (INITIALLY PRESET TO 0)  
NEW\_MINOR\_CYCLE\_FLAG  
PERIODIC PROGRAM PARAMETERS  
-STACK\_POINTER  
-DEPTH\_OF\_STACK

DATA PRODUCED

NONE

DISPATCH

THIS PROCESS CALLS ALL THE PERIODIC PROCESSES  
EXECUTED BY THE 7-2. IT IS THE LOWEST PRIORITY  
PROCESS AND IS PRE-EMPTED BY ALL INTERRUPTS.  
IT REPEATS FOR EACH MINOR CYCLE. PERIODIC PROCESS  
ARE PRIORITIZED BY THEIR POSITION IN THE STACK.

DOWHILE POWER IS ON  
• IF NEW\_MINOR\_CYCLE\_FLAG EQ 1 AND INIT\_FLAG EQ 1  
• • THEN  
• • • SET NEW\_MINOR\_CYCLE\_FLAG TO 0  
• • • DOWHILE STACK NOT EMPTY  
• • • • INITIALIZE MACHINE STATE FOR EACH PERIODIC PROCESS.  
• • • • EACH STACK ENTRY POINTS TO A BUFFER AREA CONTAINING  
• • • • INITIALIZATION PARAMETERS.  
• • • • CALL PROCESS REPRESENTED BY STACK ENTRY.  
• • ENDDO  
• ENDOIF  
ENDDO  
END

Figure 6-5



NADC-79161-40

MODULE NAME: READ

DATA REQUIRED

INPUT MESSAGE (ASSUMED TO BE LOCATED IN I.M. BUFFER) -  
TABLE OF MEMORY BUFFER LOCATIONS FOR EACH RCV ADDRESS.  
TABLE OF BUFFER LENGTHS FOR EACH RCV ADDRESS.  
INPUT MESSAGE RCV ADDRESS

DATA PRODUCED

INPUT MESSAGE IN RCV ADDRESS BUFFER

READ

THIS MODULE READS THE INPUT MESSAGE FROM THE INTERFACE  
MODULE. ONLY ASYNCHRONOUS COMMANDS ARE ACTUALLY SAVED.  
ALL OTHER MESSAGES ARE ACKNOWLEDGED BUT NOT SAVED.

```
READ RCV ADDRESS
IF RCV ADDRESS EQ X0
. INPUT X WORDS TO RCV ADDRESS TO INPUT MESSAGE BUFFER
. ACKNOWLEDGE INPUT
. CALL "COMMAND PROCESSING" MODULE
. ELSE
. ACKNOWLEDGE INPUT
ENDIF
END
```

Figure 6-6

MODULE NAME: WRITE

DATA REQUIRED

XMT MESSAGE ADDRESS  
TABLE OF BUFFER LOCATIONS FOR EACH XMT ADDRESS  
TABLE OF BUFFER LENGTHS FOR EACH XMT ADDRESS

DATA PRODUCED

OUTPUT MESSAGE TO INTERFACE MODULE

WRITE

THIS FUNCTION OUTPUTS THE SPECIFIED MESSAGE  
TO THE INTERFACE MODULE. IT WAITS FOR AN ACKNOWLEDGE  
FOR A SPECIFIED PERIOD OF TIME. NONE OCCURRING, AN  
ERROR CONDITION IS NOTED IN THE STATUS WORD.

END

Figure 6-7

MODULE NAME: COMMAND PROCESSING

DATA REQUIRED

INPUT MESSAGE (RCV ADDRESS 30)  
-COMMAND WORD  
-X HOOK POSITION  
-Y HOOK POSITION

DATA PRODUCED

NONE

COMMAND PROCESS

THIS MODULE INTERPRETS THE COMMAND ID AND CALLS  
THE APPROPRIATE SUBROUTINE TO EXECUTE THE COMMAND.

CASE COMMAND ID  
  . \*INITIALIZE NAVT\*  
  . \*TERMINATE NAVT\*  
  . \*MARK TIMER\*  
  . \*ENTER FIX\*  
  . \*DESTROY FIX\*  
  . \*ENTER FTR\*  
  . \*DESTROY FTR\*  
  . \*CENTER LAT\*  
  . \*CENTER LONG\*  
  . \*ACKNOWLEDGE CONTACT\*  
  . \*DESTROY CONTACT\*  
  . \*PROP SONAR\*  
  . \*CONTROLLER STATUS\*  
  . \*NEW MINOR CYCLE\*  
ENDCASE  
END

Figure 6-8

NADC-79161-40

Figures 6-9 through 6-23. Node 70 Operational Functions

NADC-79161-40

```
NEW MINOR CYCLE
SET NEW_MINOR_CYCLE_FLAG TO 1
END
```

Figure 6-9

```
INITIALIZE NAV
SET NAV_INIT_FLAG TO 1
END
```

Figure 6-10

```
TERMINATE NAV
SET NAV_INIT_FLAG TO 0
END
```

Figure 6-11

```
MARK TIME
THIS MODULE INITIALIZES TIME TO 0
AND STARTS THE TIME_OF_DAY COUNTER
SET TIME_OF_DAY_COUNTER TO 0
START TIME_OF_DAY_COUNTER
END
```

Figure 6-12



NADC-79161-40

END

MODULE NAME: ENTER FTP

DATA REQUIRED

X,Y HOOK POSITIONS  
FTP STACK  
POINTERS, INDICES, STACK PARAMETERS

DATA PRODUCED

NEW FTP STACK ENTRY

ENTER FTP

THIS MODULE ADDS AN ENTRY TO THE FTP STACK  
CONVERT X,Y HOOK POSITION TO U,V GRID CO-ORD  
INSERT GRID CO-ORD AT TOP OF FTP STACK  
INCREMENT NUM\_FTP

Figure 6-13

MODULE NAME: DESTROY FTP

DATA REQUIRED

X,Y HOOK POSITION  
FTP STACK  
LAST\_FTP  
FIRST\_FTP

DATA PRODUCED

MODIFIED FTP STACK

DESTROY FTP

THIS MODULE REMOVES THE SPECIFIED ENTRY FROM THE FTP STACK

SET FTP\_INDEX TO 1  
SET FIND\_FLAG TO 0  
DOWHILE FIND\_FLAG EQ 0 AND FTP\_INDEX LE NUM\_FTP  
 . IF X,Y POSITION OF HOOK "CLOSE ENOUGH" TO FTP(FTP\_INDEX)  
 . . THEN  
 . . DELETE ENTRY AND SHORTEN STACK  
 . . SET FIND\_FLAG TO 1  
 . ENDIF  
 . INCREMENT FTP\_INDEX  
ENDDO

END

Figure 6-14

NADC-79161-40

MODULE NAME: ENTER FIX

DATA REQUIRED

X,Y HOOK POSITION  
FIX FILE  
TRACK FILE  
POINTERS, INDICES, FILE PARAMETERS

DATA PRODUCED

NEW FIX FILE ENTRY

ENTER FIX

THIS MODULE ADDS AN ENTRY TO THE FIX FILE

```
APPEND EMPTY FILE ENTRY TO FIX FILE
APPEND A "1" TO NEW_FIX_FLAG VECTOR
SET TRACK_INDEX TO 1
SET FIND_FLAG TO 0
DOWHILE TRACK_INDEX LE NUM_TRACKS AND FIND_FLAG EQ 0
. CONVERT X,Y HOOK POS TO U,V GRID CO-ORDINATES
. IF U,V CO-ORD OF HOOK "CLOSE ENOUGH" TO U,V CO-ORD
.   OF TRACK(TRACK_INDEX)
.   THEN
.     SET CONTACT ID(LAST_FIX+1) TO CONTACT ID (TRACK_INDEX)
.     SET FIND_FLAG TO 1
.   ENDIF
. INCREMENT TRACK_INDEX
ENDDO
FORMAT NEW_FIX_ID AND ENTER
(FLAT EARTH APPROX)
IF FIND_FLAG EQ 1
. THEN
.   TRANSFER LAT/LONG AND GRID CO-ORD OF TRACK(TRACK_INDEX)
.   TO FIX FILE
.   SET CONTACT ID TO TRACK ID (TRACK_INDEX)
.   SET CLASSIFICATION TO "TARGET FIX" INDICATION
. ELSE
.   CONVERT HOOK POSITION TO U/V/LAT/LONG CO-ORD
.   ENTER U,V AND LAT/LONG CO-ORDINATES IN FIX FILE
.   SET CLASSIFICATION TO "FIX"
. ENDIF
SET DISPLAY DUE TO CLASSIFICATION INDICATION
END
```

Figure 6-15

NADC-79161-40

**DROP SONOBOUY**

THIS MODULE ADDS ANOTHER ENTRY TO THE SONOBOUY FILE,  
GIVING THE SONOBOUY THE PRESENT POSITION OF THE A/C

FORMAT NEW SONOBOUY ID AND ENTER IN SONOBOUY FILE  
TRANSFER GRID CO-ORDINATES OF OWN A/C TO FILE  
SET DISPLAY CUE OF NEW ENTRY TO INDICATE SONOBOUY  
SET APPROPRIATE COMPONENT OF NEW\_SONO\_FLAG TO 1  
INCREMENT NUM\_SONOBOUTS

END

Figure 6-16

**ACKNOWLEDGE CONTACT**

THIS FILE SEARCHES THE SONOBOUY FILE TO  
DETERMINE WHICH CONTACT IS BEING ACKNOWLEDGED

SET SONO\_INDEX TO 1  
SET FIND\_FLAG TO 0  
DOWHILE SONO\_INDEX LE NUM\_SONOBOUTS AND FIND\_FLAG EQ 0  
 . IF DISPLAY CUE OF SONOBOUY(SONO\_INDEX) EQ "CONTACT"  
 . . IF X,Y POSITION OF HOOK "CLOSE ENOUGH" TO SONOBOUY POSITION  
 . . . SET DISPLAY CUE OF SONOBOUY(SONO\_INDEX) TO "ACKNOWLEDGED"  
 . . . CONTACT  
 . . . SET APPROPRIATE NEW\_SONO\_FLAG COMPONENT TO 1  
 . . . SET FIND\_FLAG TO 1  
 . . ENDOF  
 . ENDOF  
 . INCREMENT SONO\_INDEX  
ENDDO  
END

Figure 6-17

**DESTROY CONTACT**

THIS MODULE REMOVES THE "ACKNOWLEDGE CONTACT"  
DESIGNATION FROM THE APPROPRIATE SONOBOUY ENTRY IN THE SONOBOUY FILE

SET SONO\_INDEX TO 1  
SET FIND\_FLAG TO 0  
DOWHILE SONO\_INDEX LE NUM\_SONOBOUTS AND FIND\_FLAG EQ 0  
 . IF X,Y HOOK POSITION "CLOSE ENOUGH" TO SONOBOUY(SONO\_INDEX)  
 . . SET DISPLAY CUE OF SONOBOUY(SONO\_INDEX) TO "SONOBOUY" DESIGNATION  
 . . SET APPROPRIATE NEW\_SONO\_FLAG COMPONENT TO 1  
 . . SET FIND\_FLAG TO 1  
 . ENDOF  
 . INCREMENT SONO\_INDEX  
ENDDO  
END

Figure 6-18

# MODULE NAME CONTACT UPDATE NADC-79161-40

## DATA REQUIRED

```

SONOROUY FILE
-SONOROUY IF
-U GRID CO-ORD
-V GRID CO-ORD
-DISPLAY CUE
-X DISPLAY CO-ORD
-Y DISPLAY CO-ORD
-TIME_OF_UPDATE
-VECTOR END POINT (X)
-VECTOR END POINT (Y)
TRACK FILE
POINTERS, INDICES, ETC

```

## DATA PRODUCED

MODIFIED SONOROUY FILE

## CONTACT UPDATE

THIS MODULE EXECUTES PERIODICALLY EVERY 90 SECONES  
AND DETERMINES WHETHER A MOVING TARGET OBJECT IN THE TRACK  
FILE IS "CLOSE ENOUGH" TO BE DETECTED BY A SIMULATED SONOROUY

```

SET TRACK_INDEX TO 1
SET SONO_INDEX TO 1
DOWHILE TRACK_INDEX LE NUM_TRACKS
.   DOWHILE SONO_INDEX LE NUM_SONOROUYS
.   .   IF SONOROUY(SONO_INDEX) "CLOSE ENOUGH" TO TARGET(TRACK_INDEX)
.   .   .   THEN
.   .   .   SET DISPLAY CUE TO "CONTACT" DESIGNATION
.   .   .   ELSE
.   .   .   SET DISPLAY CUE TO "SONOROUY" DESIGNATION
.   .   .   ENDOF
.   .   SET TIME_OF_CONTACT TO PRESENT TIME
.   .   IF SONOROUY(SONO_INDEX) INDICATES DIFAR ID AND DISPLAY CUE
.   .   .   INDICATES "CONTACT"
.   .   .   THEN
.   .   .   CALCULATE VECTOR
.   .   .   END
.   .   .   POINTS AND ENTER IN FILE
.   .   .   ELSE
.   .   .   SET VECTOR
.   .   .   END
.   .   .   POINTS TO 0
.   .   .   ENDOF
.   .   INCREMENT SONO_INDEX
.   ENDDO
.   INCREMENT TRACK_INDEX
ENDDO
SET CONTACT_UPDATE FLAG TO 1
SET ALL COMPONENTS OF NEW_SONO_FLAG TO 1
END

```

Figure 6-19



NADC-79161-40  
MODULE NAME: DESTROY FIX

DATA REQUIRED

FIX FILE  
HOOK POSITION  
PTRS, INDICES, FILE PARAMETERS

DATA PRODUCED

UPDATED FIX FILE  
DESTROY FIX  
SET FIND\_FLAG TO 0  
SET I TO 1  
DOWHILE I LE NUM\_FIXES AND FIND\_FLAG EQ 0  
 . IF HOOK "CLOSE ENOUGH" TO FIX POSITION  
 . . REMOVE ENTRY FROM FIX FILE  
 . . REMOVE COMPONENT FROM NEW\_FIX\_FLAG  
 . . SET FIND\_FLAG TO 1  
 . ENDIF  
 . INCREMENT I  
ENDDO  
END

Figure 6-20

RECENTER LAT  
SET CENTER\_OF\_DISPLAY LATITUDE COMPONENT TO DATA WORDS  
END

Figure 6-21

RECENTER LONG  
SET CENTER\_OF\_DISPLAY LONGITUDE COMPONENT TO DATA WORDS  
END

Figure 6-22

MODULE NAME NAV OUTPUT PROCESSING

DATA REQUIRED

OWN A/C PARAMETER BUFFER  
 -LATITUDE(32 BITS-DYNAMIC)  
 -LONGITUDE (32 BITS-DYNAMIC)  
 -U GRID CO-CRC (32 BITS-DYNAMIC)  
 -V GRID CO-CRC (DYNAMIC-32 BITS)  
 -RDR ALT (STATIC)  
 -RADAR ALT (STATIC)  
 -COMMAND ALT (STATIC)  
 -GROUND SPEED(STATIC)  
 -TRUE AIR SPEED (STATIC)  
 -INDICATED AIRSPEED (STATIC)  
 -COMMAND AIRSPEED (STATIC)  
 -COMMAND AIRSPEED ERROR (STATIC)  
 -MACH NO (STATIC)  
 -RANGE TO DESTINATION (DYNAMIC-32 BITS)  
 -TIME TO GO (DYNAMIC-32 BITS)  
 -ACCELERATION (DYNAMIC-16 BITS)  
 -VELOCITY (DYNAMIC-16 BITS)  
 -HEADING (DYNAMIC-16 BITS)  
 TRACK FILE  
 FIX FILE  
 SONOGRAPHY FILE  
 POINTERS, INDICES, FILE PARAMETERS, FLAGS, ETC

DATA PRODUCED

UPDATE OF OUTPUT MESSAGE BUFFERS  
 -BUF\_6 (XMT SUB ADDRESS 6 - 15 WORDS)  
 -BUF\_10 (19 WORDS)  
 -BUF\_3, BUF\_7, BUF\_11 (21 WORDS)  
 -BUF\_4, BUF\_16 (32 WORDS)  
 -BUF\_12 (21 WORDS)  
 -BUF\_13, BUF\_14, BUF\_15 (30 WORDS)  
 -BUF\_17 (24 WORDS)

Figure 6-23a

NAV OUTPUT PROCESSING

THIS MODULE EXECUTES EVERY MINOR CYCLE AND UPDATES  
THE OUTPUT BUFFERS FOR TRANSMISSION TO OTHER NODES

BLOCK 1- PROCESS OWN A/C POSITION

IF FTP\_FLAG EQ 1

. THEN

. UPDATE LAT/LONG/U/V CO-ORD USING CONSTANT INCREMENTS

. ENTER NEW CO-ORDINATES IN OWN A/C PARAMETER BUFFER

. ELSE

. UPDATE LAT/LONG/U/V CO-ORD USING CIRCULAR APPROXIMATION

. ENTER NEW CO-ORD IN OWN A/C PARAMETER BUFFER

ENDIF

ENTER NEW CO-ORD IN BUF\_3, BUF\_7, AND BUF\_11

BLOCK 2- PROCESS OWN A/C VELOCITY AND HEADING

IF FTP\_FLAG EQ 1

. THEN

. UPDATE "RANGE TO DESTINATION" AND "TIME TO GO" PARAMETERS

. ENTER PARAMETERS IN OWN A/C BUFFER

. ENTER PARAMETERS IN BUF\_12

. ELSE

. UPDATE ACCELERATION, VELOCITY, AND HEADING PARAMETERS

. ENTER PARAMETERS IN OWN A/C BUFFER

. ENTER PARAMETERS IN BUF\_6 AND BUF\_10

ENDIF

BLOCK 3- PROCESS FTP\_STACK

IF FTP\_FLAG EQ 1

. IF POSITION OF OWN A/C "CLOSE ENOUGH" TO TOP STACK ENTRY

. . THEN

. . REMOVE TOP ENTRY FROM STACK

. . DECREMENT NUM\_FTP

. . IF NUM\_FTP EQ 0

. . . THEN

. . . SET FTP\_FLAG TO 0

. . ENDIF

. ENDIF

ENDIF

BLOCK 4- PROCESS TRACK FILE

SET TRACK\_INDEX TO 1

DOWHILE TRACK\_INDEX LE NUM\_TRACKS

. INCREMENT LAT/LONG/U/V CO-ORD OF TARGETS USING CONSTANT INCREMENTS

. IF ALL THREE FIX ID PARAMETERS ARE NON-ZERO

. . THEN

. . SET APPROPRIATE COMPONENT OF NEW\_TRACK\_FLAG TO 1

. . ENDIF

. INCREMENT TRACK\_INDEX

ENDDO

SET DISPLAY CUES IN BUF\_4 AND BUF\_16 TO 0

SET I TO 1

SET J TO 1

DOWHILE I LE NUM\_TRACKS AND J LE 2

. IF NEW\_TRACK\_FLAG(I) EQ 1

. . SET NEW\_TRACK\_FLAG(I) TO 0

. . COPY PARAMETERS INTO APPROPRIATE HALF OF BUF\_4 AND BUF\_16

. . INCREMENT J

. . ENDIF

. INCREMENT I

Figure 6-23b

EN000

```

BLOCK 5- PROCESS FIX FILE
SET DISPLAY CUES IN BUF_13, BUF_14, BUF_15 TO 0
SET I TO 1
SET J TO 1
DOWHILE J LE 6 AND I LE NUM_FIXES
. IF NEW_FIX_FLAG(I) EQ 1
. . SET NEW_FIX_FLAG(I) TO 0
. . COPY FIX ENTRY(I) INTO APPROPRIATE BUFFER AREA (INDEX BY J)
. . INCREMENT J
. . INCREMENT I
. ENDIF
. INCREMENT I
EN000

```

```

BLOCK 7- PROCESS SONOROLOGY FILE
SET DISPLAY CUES IN BUF_17 TO 0
IF CONTACT_UPDATE_FLAG EQ 1
. SET I TO 1
. SET J TO 1
. DOWHILE I LE NUM_SONOROLOGY AND J LE 2
. . IF NEW_SONO_FLAG(I) EQ 1
. . . SET NEW_SONO_FLAG(I) TO 0
. . . COPY SONOROLOGY ENTRY(I) INTO APPROPRIATE HALF OF BUFFER
. . . INCREMENT J
. . . INCREMENT I
. . ENDIF
. . INCREMENT I
. EN000
ENDIF

```

```

BLOCK 7- OUTPUT BUFFERS
SET I TO 1
DOWHILE I LE NUMBER OF OUTPUT BUFFERS
. WRITE I-TH BUFFER TO INTERFACE MODULE
EN000

```

END

Figure 6-23c



TABLE 6-7. SELF-TEST MESSAGE FORMAT

WORD (16 bits)	CONTENTS
1	Node ID, Major Cycle No., Minor Cycle No.
2	Status Register 1
3	Status Register 2
4	System Configuration Identifier
5	IFPM Test Result Word
6	Old Sequence Word (last word 8 sent to Node 90)
7	Return Word (last word 8 received from Node 90)
8	New Sequence Word

#### 6.6.2 Node 82

##### 6.6.2.1 General Description

This node formats all operator inputs (Keyset depressions) and manages the tactical and situational advisory displays. For the purposes of the implementation subset, Node 82 will perform a selected number of the navigation operator functions, the steering operator functions, the display operator functions, and the MAD operator functions. These functions are activated by a Keyset depression; they either accept alphanumeric data from the operator or present a graphical tableau of the tactical situation. The operator may also position a "hook" symbol on the display in order to obtain more information about the "hooked" item or to enter a command regarding it. A list of the selected operator commands is presented in Table 6-8.

Figures 6-24 and 6-25. Node 70 Test and Recovery Function

## MODULE NAME: NAV SELF TEST

## DATA REQUIRED

RESULTS OF SELF TEST PROGRAM  
 2 STATUS WORDS  
 OLD SEQUENCE WORD (SEQUENCE WORD LAST TRANSMITTED)  
 LAST RETURN WORD INPUT FROM NODE 90

## DATA PRODUCED

1 WORD SELF TEST MESSAGE (BUF\_29)  
 -NODE ID, MAJOR CYCLE NO, MINOR CYCLE NO  
 -STATUS WORD 1  
 -STATUS WORD 2  
 -CONFIGURATION IDENT  
 -RESULTS OF SELF TEST PROGRAM  
 -OLD SEQUENCE WORD  
 -RETURN WORD  
 -NEW SEQUENCE WORD

## NAV SELF TEST

THIS MODULE EXECUTES EVERY MINOR CYCLE

SET WORD 5 OF BUF\_29 TO RESULT OF SELF TEST PROGRAM  
 IF WORD 5 INDICATES A FAILURE

```

. THEN
.   IF FAILURE RECOVERABLE
.   . THEN
.   .   CALL APPROPRIATE RECOVERY ROUTINE
.   .   SET STATUS WORDS AND CONFIGURATION ID TO INDICATE NEW
.   .   CONFIGURATION
.   . ELSE
.   .   ATTEMPT TO ACTIVATE SHUTDOWN MECHANISM
.   . ENDIF
. ELSE
.   IF NEW MAJOR CYCLE
.   . INCREMENT MAJOR CYCLE NO.
.   . SET MINOR CYCLE NO. TO 0
.   . ENDIF
.   SET WORD 1 OF BUF_29 TO NODE ID, MAJOR CYCLE NO, MINOR CYCLE NO
.   INCREMENT MINOR CYCLE NO.
.   COPY STATUS WORD 1 INTO WORD 2 OF BUF_29
.   COPY STATUS WORD 2 INTO WORD 3 OF BUF_29
.   SET WORD 4 OF BUF_29 TO CONFIGURATION ID
.   SET WORD 5 OF BUF_29 TO LAST WORD 5 TRANSMITTED FROM BUF_29
.   SET WORD 7 OF BUF_29 TO LAST WORD 8 RECEIVED FROM NODE 90
.   SET WORD 8 OF BUF_29 TO A NEW SEQUENCE WORD
.   WRITE BUF_29 TO I.M.
. ENDIF

```

END

Figure 6-24

## CONTROLLER STATUS

THIS MODULE RETURNS THE LAST SELF TEST MESSAGE RECEIVED FROM NODE 90  
 IN RESPONSE TO A REQUEST FROM THE BACKUP/MONITOR FOR NODE 90

WRITE LAST SELF TEST MESSAGE RECEIVED FROM NODE 90 TO IM

END

Figure 6-25.

TABLE 6-8. NODE 82 COMMAND INPUTS

COMMAND	FUNCTION
Display Init.	Initializes display; North UP reference
Configure	Sets configuration for SAD's and TAC/NAV display
Alphanumeric	All text and text control characters
Clear Tactical Plot	Removes fix symbols, reference marks, and DIFAR vectors
Acknowledge Contact	Indicates acknowledged contact on sonobuoy
Reference Mark	Displays reference mark at hooked position
Clear Text	Clears all text displayed on TAC/NAV display
Recenter	Recenters display on hook
NAV Init.	Initializes simulated NAV system
Mark Time	Starts system clock
Contact Alert	Displays contact symbol at sonobuoy position
Read Latitude/Longitude	Displays latitude/longitude of hooked point
Enter Fly-to Point	Causes simulated aircraft to fly to hooked position
Interrogate	Displays track information for selected target
Enter Fix	Enters fix at hooked position
Drop Sonobuoy	Displays sonobuoy at own-aircraft position
Destroy Fix	Destroys fix at hooked position
Destroy FTP	Destroys FTP at hooked position
Terminate NAV	Terminates navigation program
MAD Reinit.	Resets MAD subsystem to initial condition
MAD Display	Initiates MAD trace with annotation
Activate FRP	Initiates feature recognition processing
MAD Disable	Terminates MAD processing



### 6.6.2.2 I/O Description

The input and output messages making up the Node 82 interface in the implementation subset are shown in Tables 6-9 and 6-10, respectively. Asynchronous message formats are dependent upon the particular command they represent and are described by the particular software module that executes them. Formats and data content for the messages to Node 82 are shown in Table 6-11.

### 6.6.2.3 Software Functional Description

6.6.2.3.1 Control Modules. The control module requirements for Node 82 are identical to those of Node 70. Figures 6-26 through 6-31 describe the software executive functions.

6.6.2.3.2 Operational Modes. These modules format data for display and output those data periodically to Node 81 for display. They operate on the files shown in Table 6-12. These files contain the same information as their counterparts in Node 70. Figures 6-32 through 6-60 describe the operational modules.

6.6.2.3.3 Test and Reconfiguration Modules. Node 82 executes a self-test program every minor cycle and reports the results to Node 90 by means of a self-test output message. Figure 6-61 describes this module.

6.6.2.3.4 Node 82 Description. Node 82 represents the operator display portion of the system. This node receives data from the bus and displays it without modification. It receives data only from Node 81 according to the message formats specified in Table 6-11. The actual implementation of Node 82 is dependent on the display hardware available and is presently being implemented by 502. Formal documentation is presently unavailable. For this reason, Node 82 is not further described. It does, however, perform the same self-test function as Node 81 every minor cycle and sends the resultant message to Node 90.

TABLE 6-9. NODE 82 INPUT MESSAGES

	SRC NODE	RCV SUB ADR	WORDS PER SECOND	PERIOD IN MSEC	CYC CODE	WORD COUNT
***						
*	21	30	6	0		3
*	30	6	400	40	0	20
*	30	7	3600	160	0	32
*		8			0	32
*		9			0	32
*		10			0	32
*		11			0	32
*		12			0	32
*	30	30	3	0		3
*	70	16	1240	50	0	19
*		17			0	21
*		18			0	21
*	70	19	1500	2000	0	30
*		20			0	30
*		21			0	30
*	70	22	320	1000	1	32
*	70	23	100	2000	2	11
*	70	30	16	0		16
*	90	30		0		3
*	108	24	8	1000	4	8
*	123	30	3	0		3
*	124	30	40	0		20
*	125	30	40	0		20
*	126	30	40	0		20
						1

TABLE 6-10. NODE 82 OUTPUT MESSAGES

	DEST. NODE	XMT SUB ADR	WORDS PER SECOND	PERIOD IN MSEC	CYC CODE	WORD COUNT
***						
*	21	30	6	0		3
*	30	30	3	0		3
*	70	30	14	0	4	14
*	70	30	25	0		5
*	81	5	4000	500	0	32
*		6			0	32
*		7			0	32
*		8			0	32
*		9			0	32
*		10			0	32
*		11			1	16
*	90	29	160	50	0	8
						1

TABLE 6-11. NODE 82 OUTPUT MESSAGES TO DISPLAY (NODE 81)

[illegible]



NADC-79161-40

Figures 6-26 through 6-31. Node 82 Executive Functions

NADC-79161-40  
MODULE NAME: INITIALIZE DISPLAY SYSTEM

DATA REQUIRED

NONE

DATA PRODUCED

NONE

INITIALIZE DISPLAY SYSTEM

THIS PROCESS IS ACTUATED BY A POWER ON CONDITION (IF POSSIBLE). IT LOADS PROGRAMS AND PRESET DATA DISPLAY SYSTEM FROM THE BUS AND SETS THE NECESSARY INITIAL VALUES. IT ALSO INITIALIZES THE INTERRUPT STRUCTURE AND SCHEDULES THE OUTPUT PROCESSING AND SELF TEST MODULES. FINALLY, IT CALLS THE "DISPATCH" MODULE TO INITIATE PROCESSING.

END

Figure 6-26

MODULE NAME: INTERRUPT HANDLER

DATA REQUIRED

TYPE OF EVENT (EXTERNAL OR BUS INPUT)

DATA PRODUCED

NONE

INTERRUPT HANDLER

THIS FUNCTION SAVES THE STATE OF THE MACHINE AND TRANSFERS CONTROL TO THE PROPER INTERRUPT RESPONSE MODULE

LOCK OUT ALL FURTHER INTERRUPTS  
SAVE PROGRAM CTR  
SAVE STACK POINTERS  
SAVE REGISTER CONTENTS  
CASE INTERRUPT TYPE (EXTERNAL OR BUS INPUT)  
  . "EXTERNAL INTERRUPT"  
  . "READ"  
ENDCASE  
RESTORE MACHINE STATE  
ENABLE INTERRUPTS  
TRANSFER TO PROGRAM CTR

END

Figure 6-27

NADC-79161-40

MODULE NAME: DISPATCH

DATA REQUIRED

INIT\_FLAG (INITIALLY PRESET TO 0)  
NEW\_MINOR\_CYCLE\_FLAG  
PERIODIC PROGRAM PARAMETERS  
-STACK\_POINTER  
-DEPTH\_OF\_STACK

DATA PRODUCED

NONE

DISPATCH

THIS PROCESS CALLS ALL THE PERIODIC PROCESSES  
EXECUTED BY THE Z-2. IT IS THE LOWEST PRIORITY  
PROCESS AND IS PRE-EMPTED BY ALL INTERRUPTS.  
IT REPEATS FOR EACH MINOR CYCLE. PERIODIC PROCESS  
ARE PRIORITIZED BY THEIR POSITION IN THE STACK.

DOWHILE POWER IS ON  
• IF NEW\_MINOR\_CYCLE\_FLAG EQ 1 AND INIT\_FLAG EQ 1  
• • THEN  
• • SET NEW\_MINOR\_CYCLE\_FLAG TO 0  
• • DOWHILE STACK NOT EMPTY  
• • • INITIALIZE MACHINE STATE FOR EACH PERIODIC PROCESS.  
• • • EACH STACK ENTRY POINTS TO A BUFFER AREA CONTAINING  
• • • INITIALIZATION PARAMETERS.  
• • • CALL PROCESS REPRESENTED BY STACK ENTRY.  
• • ENDDO  
• ENDIF  
ENDDO  
END

Figure 6-28

MODULE NAME: READ

DATA REQUIRED

INPUT MESSAGE (ASSUMED TO BE LOCATED IN I.M. BUFFER)  
TABLE OF MEMORY BUFFER LOCATIONS FOR EACH RCV ADDRESS.  
TABLE OF BUFFER LENGTHS FOR EACH RCV ADDRESS.  
INPUT MESSAGE RCV ADDRESS

DATA PRODUCED

INPUT MESSAGE IN RCV ADDRESS BUFFER

Figure 6-29

NADC-79161-40

READ

THIS MODULE INPUTS MESSAGES FROM THE INTERFACE MODULE

READ RCV ADDRESS

IF SOURCE NODE EQ NODE 21

• THEN

• ACKNOWLEDGE INPUT

• ELSE

• IF RCV ADDRESS EQ 30

• THEN

• INPUT THREE WORDS TO RCV ADDRESS 30 MESSAGE BUFFER

• ACKNOWLEDGE INPUT

• CALL "COMMAND PROCESSING" MODULE

• ELSE

• TRANSFER MESSAGE TO APPROPRIATE INPUT BUFFER

• ACKNOWLEDGE INPUT

• ENDIF

ENDIF

END

Figure 6-30

MODULE NAME: WRITE

DATA REQUIRED

XMT MESSAGE ADDRESS

TABLE OF BUFFER LOCATIONS FOR EACH XMT ADDRESS

TABLE OF BUFFER LENGTHS FOR EACH XMT ADDRESS

DATA PRODUCED

OUTPUT MESSAGE TO INTERFACE MODULE

WRITE

THIS FUNCTION OUTPUTS THE SPECIFIED MESSAGE TO THE INTERFACE MODULE. IT WAITS FOR AN ACKNOWLEDGE FOR A SPECIFIED PERIOD OF TIME, NONE OCCURRING ERROR CONDITION IS NOTED IN THE STATUS WORD.

END

Figure 6-31



TABLE 6-12. NODE 82 DATA FILES

TRACK FILE (Contains current position of all targets)

- Track ID
- Fix 1 ID
- Fix 2 ID
- Fix 3 ID
- Target Latitude (2 words)
- Target Longitude (2 words)
- Target Grid Coordinates (U) (2 words)
- Target Grid Coordinates (V) (2 words)
- Bearing
- Target Velocity
- Target Classification
- Display Cue

FIX FILE

- Contact ID
- Fix ID
- Fix Latitude (2 words)
- Fix Longitude (2 words)
- Fix Grid Coordinates (U) (2 words)
- Fix Grid Coordinates (V) (2 words)
- Classification
- Display Cue
- X Display Coordinates
- Y Display Coordinates
- Time of Fix (2 words)

SONOBUOY FILE

- Sonobuoy ID
- Grid Coordinates (U) (2 words)
- Grid Coordinates (V) (2 words)
- Vector Endpoint (U) (2 words)
- Vector Endpoint (V) (2 words)
- Time of Update (2 words)
- Display Cue
- X Display Coordinates
- Y Display Coordinates

REFERENCE MARK FILE

- Reference Mark ID
- X Display Coordinates
- Y Display Coordinates

?

NADC-79161-40

Figures 6-32 through 6-60. Node 81 Operational Functions

NADC-79161-40

MODULE NAME: DISPLAY OUTPUT PROCESSING

DATA REQUIRED

TRACK FILE  
FIX FILE  
SONOBOUY FILE  
OWN A/C PARAMETER BUFFER  
TEXT FILE  
PTRS, INDICES, FILE PARAMETERS, ETC  
REFERENCE MARK FILE

DATA PRODUCED

UPDATES OF OUTPUT BUFFERS  
BUF\_3 (TRACK FILE- 30 WORDS)  
BUF\_9 (HOOK, A/C, AND VECTOR SYMBOLS - 30 WORDS)  
BUF\_10 (FIXES, SONOBOUYS, AND REFERENCE MARKS - 30 WORDS)  
BUF\_11 (TEXT FILE - 16 WORDS)

DISPLAY OUTPUT PROCESSING

THIS MODULE UPDATES TRANSMIT SUB-ADDRESSES 8, 9, 10, AND 11  
IN THE FORMAT DESCRIBED IN TABLE 6-1, AND TRANSMITS THEM EVERY  
MINOR CYCLE

BLOCK 1A- HOOK POSITION

INSERT X AND Y DISPLAY CO-ORDINATES OF HOOK IN WORDS 1 AND 2 OF BUFFER

BLOCK 1B- A/C POSITION

READ A/C POSITION CO-ORD FROM OWN A/C PARAMETER BUFFER  
CONVERT TO X,Y DISPLAY CO-ORD  
READ HEADING FROM OWN A/C PARAMETER BUFFER  
CONVERT TO SIN, COS DISPLAY FORMAT  
INSERT POSITION AND HEADING DATA IN WORDS 3,4,5,6 OF BUF\_9

BLOCK 1C - VECTOR INFORMATION (BEARING LINES FROM DIFAR SONOBOUYS)

SET I TO 1

SET WORDS 7 THRU 30 OF BUF\_9 TO 0

SET J TO 7

DOWHILE I LE NUM\_SONOBOUYS AND J LE 30

. IF NEW\_SONO\_FLAG(I) NE 0  
. . SET NEW\_SONO\_FLAG(I) TO 0  
. . IF VECTOR\_ENDPOINTS FOR ENTRY(I) NE 0,0  
. . . CONVERT VECTOR\_ENDPOINTS IN ENTRY(I) TO X,Y DISPLAY  
. . . CO-ORDINATES  
. . . INSERT IN BUF\_9 AS VECTOR\_ENDPOINT DATA (WORDS J+2, J+3)  
. . . COPY SONOBOUY POSITION CO-ORD INTO BUF\_9 AS VECTOR START POINT  
. . . DATA  
. . . (WORDS J, J+1)  
. . . SET VECTOR\_ENDPOINTS IN SONOBOUY FILE TO 0,0  
. . . INCREMENT J BY 4  
. . ENDOF  
. ENDOF  
. INCREMENT I  
ENDDO

Figure 6-32a

NADC-79161-40

```

BLOCK 2- UPDATE BUF_8 (TRACK INFO)
SET BUF_8 TO 0
SET I TO 1
SET J TO 1
DOWHILE I LE NUM_TRACKS
. IF DISPLAY CUE OF TRACK FILE ENTRY(I) NE 0
. . CONVERT POSITION CO-ORD OF TRACK(I) TO X,Y DISPLAY CO-ORD
. . SET DISPLAY CUE TO 0
. . CONVERT TIME-OF-UPDATE TO ALPHANUMERICS
. . INSERT CO-ORD AND TIME OF UPDATE IN WORDS J, J+1, AND J+2 OF
. . BUF_8
. . INCREMENT J BY 3
. ENDF
. INCREMENT I
ENDDO

BLOCK 3- UPDATE BUF_10 (FIX, SONOBUOY, AND REFERENCE MARKS)
SET J TO 1

BLOCK 3A (SONOBUOY FILE)

SET I TO 1
SET BUF_10 TO 0

DOWHILE I LE NUM_SONOBBOUYS AND J LE 30
. IF NEW_SONO_FLAG(I) NE 0
. . SET NEW_SONO_FLAG(I) TO 0
. . CONVERT SONOBUOY ID IN ENTRY(I) TO MEMORY ADDRESS (40 THROUGH
. . 127)
. . INSERT MEMORY ADDRESS IN WORD J
. . INSERT X,Y DISPLAY CO-ORDINATES IN WORDS J AND J+1
. . CONVERT DISPLAY CUE TO SYMBOL CODE
. . INSERT SYMBOL CODE IN WORD J+2
. . IF DISPLAY CUE INDICATES SONOBUOY
. . . THEN
. . . IF DIFAR TYPE
. . . . THEN
. . . . . SET ID CHAR 1 TO "0"
. . . . . ELSE
. . . . . SET ID CHAR 1 TO "L"
. . . . ENDF
. . . CONVERT SONOBUOY ID TO TWO DIGIT ALPHANUMERIC CODE
. . . INSERT ALPHANUMERICS IN ID CHAR 2,3
. . . SET TIME ALPHANUMERICS TO BLANKS
. . . ELSE
. . . (CONTACT OR ACKNOWLEDGED CONTACT)
. . . CONVERT CONTACT ID TO TWO DIGIT ALPHANUMERIC CODE
. . . SET ID CHARACTERS 1 AND 2 IN BUF_10 TO ID
. . . SET ID CHAR 3 TO BLANK
. . . SET TIME ALPHANUMERICS TO TIME-OF-UPDATE
. . . ENDF
. . INCREMENT J BY 6
. ENDF
. INCREMENT I
ENDDO
SET I TO 1
DOWHILE I LE NUM_FIXES AND J LE 30
. IF NEW_FIX_FLAG(I) NE 0
. . SET NEW_FIX_FLAG(I) TO 0
. . CONVERT FIX ID TO MEMORY ADDRESS
. . INSERT MEMORY ADDRESS IN WORD J
. . INSERT X,Y DISPLAY CO-ORD IN WORDS J AND J+1
. . CONVERT DISPLAY CUE TO SYMBOL CODE
. . INSERT SYMBOL CODE IN WORD J+2
. . CONVERT FIX ID TO 2 DIGIT ALPHANUMERIC
. . SET ID CHARACTERS 1 AND 2 TO FIX ID IN WORDS J+2, J+3
. . SET ID CHAR 3 TO BLANK
. . SET TIME ALPHANUMERICS TO TIME OF FIX (WORDS J+4, J+5)
. . INCREMENT J BY 6

```

Figure 6-32b



NADC-79161-40

```
. ENDOIF
. INCREMENT I
ENDDO
SET I TO 1
DOWHILE I LE NUM_REFERENCES AND J LT 30
. IF NEW_REF_FLAG(I) NE 0
. . SET NEW_REF_FLAG(I) TO 0
. . COPY X,Y DISPLAY CO-ORD INTO WORDS J AND J+1
. . SET ID CHAR 1 TO "R"
. . SET ID CHARS 2 AND 3 TO REPRESENT "I"
. . SET TIME ALPHANUMERICS TO BLANKS
. . INCREMENT J BY 6
. ENDOIF
. INCREMENT I
ENDDO

BLOCK 4- UPDATE BUF_11 (TEXT)
IF TAC_NAV_DISPLAY FLAG NE 0
. THEN
. . SET I TO LAST_CHAR_OUT
. . SET J TO LAST_X_POSITION
. . SET K TO 1
. . SET BUF_11 TO ALL BLANKS
. . DOWHILE I LE LAST_CHAR_IN AND J LE 28
. . . IF TEXT_CHAR(I) EQ CONTROL_CHAR
. . . . THEN
. . . . . CASE CONTROL_CHARACTER
. . . . . . CARRIAGE_RETURN
. . . . . . - SET J TO 29
. . . . . . BACKSPACE
. . . . . . - SET J TO MAX(J-1,1)
. . . . . . -SET K TO MAX(K-1,1)
. . . . . . -WRITE BLANK_CHAR IN KTH BUFFER_POSITION
. . . . . . ERASE_LINE
. . . . . . -ENTER BLANKS IN BUF_11
. . . . . . -SET J TO 29
. . . . . . -DECREMENT Y_POSITION BY DEL_Y
. . . . . ENDCASE
. . . . ELSE
. . . . . COPY_CHAR_TO_BUFFER_POSITION_K
. . . . . INCREMENT J
. . . . . INCREMENT LAST_X_POSITION
. . . . . INCREMENT K
. . . . ENDOIF
. . . INCREMENT I
. . ENDDO
. . INSERT LAST_X_POSITION, Y_POSITION IN WORDS 1 AND 2 OF BUF_11
. . IF J EQ 29
. . . THEN
. . . . SET LAST_X_POSITION TO MIN_X
. . . . INCREMENT Y_POSITION BY DEL_Y
. . . . IF Y_POSITION GT MAX_Y
. . . . . THEN
. . . . . . SET Y_POSITION TO MIN_Y
. . . . . ENDOIF
. . . . ELSE
. . . . . SET LAST_X_POSITION TO LAST_X_POSITION+J X DEL_X
. . . . ENDOIF
. . ENDOIF
END
```

Figure 6-32c

NADC-79161-40

MODULE NAME: COMMAND PROCESSING (DISPLAY)

DATA REQUIRED

COMMAND ID  
FIX FILE  
TRACK FILE  
SONOBUOY FILE  
REFERENCE MARK FILE  
TEXT FILE  
POINTERS, INDICES, FILE PARAMETERS  
CENTER\_OF\_DISPLAY VARIABLE  
TRACK\_INTERROGATE\_NUMBER

DATA PRODUCED

NONE

COMMAND PROCESSING (DISPLAY)

THIS MODULE IDENTIFIES THE INPUT COMMAND AND CALLS THE PROPER SERVICE ROUTINE. IT IS ACTIVATED BY THE "READ" COMMAND UPON RECEPTION OF AN ASYNCHRONOUS MESSAGE.

CASE COMMAND ID  
  . \$DISPLAY INIT\$  
  . \$DISPLAY CONFIGURES  
  . \$ALPHANUMERIC\$  
  . \$CLEAR TACTICAL PLOTS  
  . \$ACKNOWLEDGE CONTACT\$  
  . \$REFERENCE MARK\$  
  . \$CLEAR TEXT\$  
  . \$RECENTERS  
  . \$NAV INIT\$  
  . \$MARK TIME\$  
  . \$READ LAT/LONG\$  
  . \$ENTER FTP\$  
  . \$TRACK INTERROGATE\$  
  . \$ENTER FIX\$  
  . \$DROP SONOBUOY\$  
  . \$DESTROY FIX\$  
  . \$DESTROY FTP\$  
  . \$TERMINATE NAV\$  
  . \$MAD RE-INIT\$  
  . \$MAD DISPLAY\$  
  . \$ACTIVATE FRP\$  
  . \$MAD DISABLE\$  
  . \$COMM INIT\$  
  . \$COMM TERM\$  
  . \$NEW MINOR CYCLE\$  
ENDCASE

END

Figure 6-33

AD-A072 427

NAVAL AIR DEVELOPMENT CENTER WARMINSTER PA COMMUNICA--ETC F/G 5/2  
BASIC LABORATORY ARCHITECTURE PLAN.(U)

MAY 79 S GREENBERG, C JOECKEL, R MEJZAK

IDWA-45-78-04

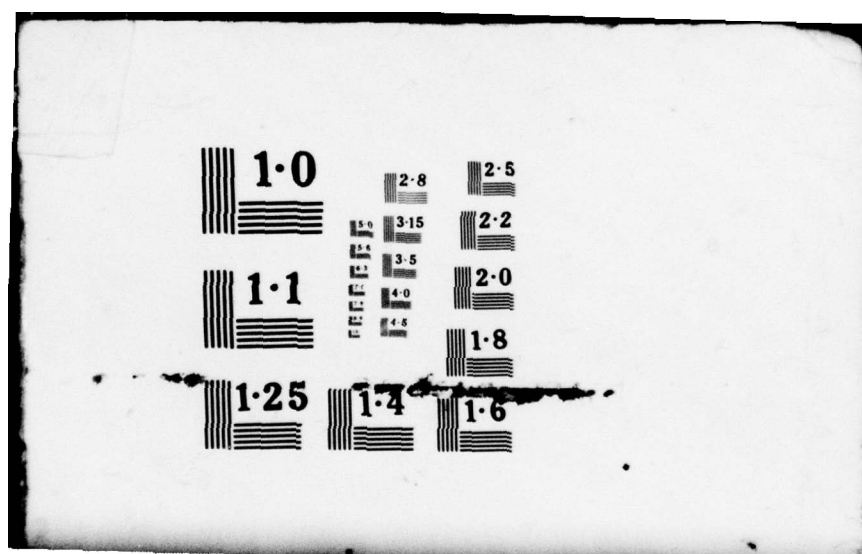
UNCLASSIFIED

NADC-79161-40

NL

3 OF 4  
AD  
A072427







NADC-79161-40

DISPLAY INIT

SET DISPLAY\_INIT\_FLAG TO 1  
SEND "NORTH UP MODE INITIATE" CMD TO DISPLAY NODE 92  
WRITE "NAV INIT" CMD TO NODE 70  
SET CENTER\_OF\_DISPLAY TO PRESET LAT/LONG  
INITIALIZE FILE STRUCTURES  
SET TRACK\_INTERROGATE\_NUMBER TO 1

END

DISPLAY CONFIGURE

SET TAC/NAV\_DISPLAY\_FLAG TO 1  
SET SAD\_1\_RE\_INIT\_FLAG TO 1  
SET SAD\_3\_TEXT\_FLAG TO 1

END

Figure 6-34

ALPHANUMERIC

THIS MODULE ACCEPTS TEXT OR CONTROL CHARACTERS AND  
APPENDS THEM TO THE TEXT FILE

APPEND CHARACTER TO TEXT FILE  
INCREMENT LAST\_CHAR\_IN

END

Figure 6-35

CLEAR TACTICAL PLOT

THIS MODULE CLEARS ALL FIXES, REFERENCE MARKS, VECTORS, AND TEXT  
FROM THE TAC/NAV DISPLAY

RE-INITIALIZE FIX FILE  
PRE-INITIALIZE TEXT FILE  
RE-INITIALIZE REFERENCE MARK FILE  
SET NEW\_FIX\_FLAG VECTOR TO 0 LENGTH  
SET NEW\_REF\_FLAG VECTOR TO 0 LENGTH  
SET LAST\_CHAR\_IN TO 1  
SET ALL COMPONENTS OF NEW\_SONO\_FLAG TO 1  
WRITE DEDICATED MEMORY ERASE COMMAND TO NODE 81  
WRITE NON DEDICATED MEMORY ERASE COMMAND TO NODE 81

END

Figure 6-36

NADC-79161-40

ACKNOWLEDGE CONTACT

WRITE "ACKNOWLEDGE CONTACT" COMMAND TO NODE 70  
 DATA WORDS ARE HOOK X,Y POSITIONS

END

Figure 6-37

REFERENCE MARK

ENTER REFERENCE MARK ID IN REFERENCE MARK FILE  
 ENTER X DISPLAY CO-ORD  
 ENTER Y DISPLAY CO-ORD  
 APPEND A "1" TO NEW\_REF\_FLAG VECTOR

END

Figure 6-38

RECENTER

CALCULATE NEW CENTER\_OF\_DISPLAY (LAT/LONG) FROM  
 HOOK POSITION AND OLD CENTER\_OF\_DISPLAY  
 SEND "RECENTER LAT" COMMAND TO NODE 70  
 (DATA WORDS ARE LATITUDE)  
 SEND "RECENTER LONG" COMMAND TO NODE 70  
 (DATA WORDS ARE LONGITUDE)  
 SET I TO 1  
 DOWHILE I LE NUM\_FIXES  
 . UPDATE X,Y DISPLAY CO-ORD OF ENTRY(I)  
 . SET NEW\_FIX\_FLAG(I) TO 1  
 ENDDO  
 SET I TO 1  
 DOWHILE I LE NUM\_SONOBUOYS  
 . UPDATE X,Y DISPLAY CO-ORD OF ENTRY(I)  
 . SET NEW\_SONOB\_FLAG(I) TO 1  
 ENDDO  
 SET I TO 1  
 DOWHILE I LE NUM\_REFERENCES  
 . UPDATE X,Y DISPLAY CO-ORD OF ENTRY(I)  
 . SET NEW\_REF\_FLAG(I) TO 1  
 ENDDO  
 WRITE "DEDICATED MEMORY ERASE" COMMAND TO NODE 81  
 WRITE "NON-DEDICATED MEMORY ERASE" COMMAND TO NODE 81

END

Figure 6-39

NADC-79161-40

CLEAR TEXT

RE-INITIALIZE TEXT FILE

SET LAST\_CHAR\_IN TO 0

WRITE "NON-DEDICATED MEMORY ERASE" COMMAND TO DISPLAY-NODE 81

END

Figure 6-40

NAV INIT

WRITE "NAV INIT" COMMAND TO NODE 70

END

Figure 6-41

COMM INIT

WRITE "COMM INIT" COMMAND TO NODE 21

USE PRESET INITIALIZATION DATA FOR DATA WORDS 1 AND 2

END

Figure 6-42

COMM TERM

WRITE "COMM TERM" COMMAND TO NODE 21

END

Figure 6-43

NADC-79161-40

MARK TIME

WRITE "MARK TIME" COMMAND TO NODE 70

END

Figure 6-44

READ LAT/LONG

APPROXIMATE LAT/LONG FROM HOOK POSITION AND CENTER\_OF\_DISPLAY

CONVERT TO ALPHANUMERICS

APPEND ALPHANUMERICS TO TEXT FILE

UPDATE LAST\_CHAR\_IN

END

Figure 6-45

ENTER FTP

WRITE "ENTER FTP" COMMAND TO NODE 70

(HOOK POSITIONS ARE DATA WORDS)

ENTER FTP AND HOOK POSITIONS IN REFERENCE MARK FILE

APPEND A "1" TO NEW\_REF\_FLAG VECTOR

END

Figure 6-46



## TRACK INTERROGATE

THIS ROUTINE BRINGS THE TRACK SYMBOL UP ON THE TAC/NAV DISPLAY AND PROVIDES POSITION, HEADING, AND VELOCITY INFORMATION ON THE SAD\_3 DISPLAY. A SEQUENCE OF "TRACK INTERROGATE" COMMANDS WILL GIVE INFORMATION ON EACH TARGET IN TURN. AFTER THE LAST TARGET IS INTERROGATED, THE COMMAND WILL RETURN OPERATION TO NORMAL

```

IF TRACK_INTERROGATE_NUMBER EQ NUM_TRACKS
. THEN
. SET TRACK_INTERROGATE_NUMBER TO 0
. ELSE
. INCREMENT TRACK_INTERROGATE_NUMBER BY 1
ENDIF
IF TRACK_INTERROGATE_NUMBER NE 0
. THEN
. SET DISPLAY CUE OF TRACK ENTRY(TRACK_INTERROGATE_NUMBER) TO "TARGET"
. SET UP TARGET TABLEAU IN SAD_3 OUTPUT BUFFER
. -TARGET ID ----
. -LATITUDE ----
. -LONGITUDE ----
. -U GRID CO-ORD ----
. -V GRID CO-ORD ----
. -BEARING ----
. -VELOCITY ----
. -CLASSIFICATION ----
. CONVERT TARGET PARAMETERS IN TRACK FILE
. ENTRY(TRACK_INTERROGATE_NUMBER)
. TO ALPHANUMERICS AND ENTER IN OUTPUT BUFFER
. ELSE
. SET SAD_3_TEXT_FLAG TO 1
ENDIF
END

```

Figure 6-47

## ENTER FIX

```

SEND "ENTER FIX" COMMAND TO NODE 70
(DATA WORDS ARE X,Y HOOK POSITIONS)
END

```

Figure 6-48

## DROP SONOBUOY

```

WRITE "DROP SONOBUOY" COMMAND TO NODE 70
(NO DATA WORDS)
END

```

Figure 6-49

NADC-79161-40

DESTROY FIX

```
SEND "DESTROY FIX" COMMAND TO NODE 70
(DATA WORDS ARE X,Y HOOK POSITIONS)
SET I TO 1
SET FIND_FLAG TO 0
DOWHILE I LE NUM_FIXES AND FIND_FLAG EQ 0
. IF HOOK POSITION "CLOSE ENOUGH" TO FIX POSITION
. . SET FIND_FLAG TO 1
. . REMOVE ENTRY(I) FROM FILE
. . REMOVE COMPONENT(I) FROM NEW_FIX_FLAG VECTOR
. . SET ALL NEW_FIX_FLAG COMPONENTS TO 1
. . SET ALL NEW_SONO_FLAG COMPONENTS TO 1
. . SET ALL NEW_REF_FLAG COMPONENTS TO 1
. . WRITE "ERASE DEDICATED MEMORY" COMMAND TO NODE 81
. ENDF
. INCREMENT I
ENDDO
END
```

Figure 6-50

DESTROY FTP

```
SEND "DESTROY FTP" COMMAND TO NODE 70
(DATA WORDS ARE X,Y HOOK POSITIONS)
SET I TO 1
SET FIND_FLAG TO 0
DOWHILE I LE NUM_REFERENCES AND FIND_FLAG EQ 0
. IF HOOK POSITION "CLOSE ENOUGH" TO FTP POSITION
. . SET FIND_FLAG TO 1
. . REMOVE ENTRY(I) FROM FILE
. . REMOVE COMPONENT(I) FROM NEW_REF_FLAG VECTOR
. . SET ALL NEW_REF_FLAG COMPONENTS TO 1
. . SET ALL NEW_SONO_FLAG COMPONENTS TO 1
. . SET ALL NEW_FIX_FLAG COMPONENTS TO 1
. . WRITE "ERASE DEDICATED MEMORY" COMMAND TO NODE 81
. ENDF
. INCREMENT I
ENDDO
END
```

Figure 6-51

TERMINATE NAV

```
SEND "TERMINATE NAV" COMMAND TO NODE 70
NO DATA WORDS
END
```

Figure 6-52

NADC-79161-40

MAD RE\_INIT

SEND "MAD RE\_INIT" COMMAND TO NODE 70  
(NO DATA WORDS)

END

Figure 6-53

MAD DISPLAY

SEND "MAD DISPLAY" COMMAND TO NODE 70  
(NO DATA WORDS)

END

Figure 6-54

ACTIVATE FRP

SEND "ACTIVATE FRP" COMMAND TO NODE 70  
(NO DATA WORDS)

END

Figure 6-55

MAD DISABLE

SEND "MAD DISABLE" COMMAND TO NODE 70  
(NO DATA WORDS)

END

STRUCTURE

NEW MINOR CYCLE

SET NEW\_MINOR\_CYCLE\_FLAG TO 1

END

Figure 6-56

MODULE NAME: SAD1 UPDATE

## DATA REQUIRED

FTP\_FLAG  
 SAD\_1\_RE\_INIT\_FLAG  
 OWN A/C PARAMETER BUFFER

## DATA PRODUCED

SAD1 OUTPUT BUFFER  
 -16 LINES (64 ALPHANUMERIC CHARACTERS EACH)

## SAD1 UPDATE

THIS MODULE EXECUTES APPROXIMATELY EVERY SECOND

```

IF FTP_FLAG EQ 1
  THEN
    IF SAD_1_RE_INIT_FLAG EQ 1
      (INITIALIZE SAD1 BUFFER FOR STRAIGHT LINE FLIGHT PATH)
      SET LINE 1 TO "NAVIGATION INFORMATION"
      SET LINE 2 TO BLANKS
      SET LINE 3 TO "STRAIGHT LINE FLIGHT PATH"
      SET LINE 4 TO ALL BLANKS
      SET LINE 5 TO "LATITUDE: _____ DEG; _____ MIN"
      SET LINE 6 TO "LONGITUDE: _____ DEG; _____ MIN"
      SET LINE 7 TO "HEADING: _____ DEG; _____ MIN"
      SET LINE 8 TO "AZIMUTH: _____ DEG; _____ MIN"
      SET LINE 9 TO "BEARING TO DESTINATION: _____ DEG"
      SET LINE 10 TO "RANGE TO DESTINATION: _____ NM"
      SET LINE 11 TO "TIME TO GO: _____ MIN"
      SET LINE 12 TO BLANKS
      SET LINE 13 TO BLANKS
      SET LINE 14 TO BLANKS
      SET LINE 15 TO "TIME: _____ HRS; _____ MIN; _____ SEC"
      SET SAD_1_RE_INIT_FLAG TO 0
    ENDIF
    READ APPROPRIATE INFORMATION FROM OWN A/C BUFFER
    CONVERT TO ALPHANUMERICS
    UPDATE PROPER LINES
  ELSE
    IF SAD_1_RE_INIT_FLAG EQ 1
      (INITIALIZE SAD1 BUFFER FOR CIRCULAR FLIGHT PATH)
      SET LINE 1 TO "NAVIGATION INFORMATION"
      SET LINE 2 TO BLANKS
      SET LINE 3 TO "CIRCULAR FLIGHT PATH"
      SET LINE 4 TO BLANKS
      SET LINE 5 TO "LATITUDE: _____ DEG; _____ MIN"
      SET LINE 6 TO "LONGITUDE: _____ DEG; _____ MIN"
      SET LINE 7 TO "U POSITION: _____ NM"
      SET LINE 8 TO "V POSITION: _____ NM"
      SET LINE 9 TO "BARO ALT: _____ FT"
      SET LINE 10 TO "RADAR ALT: _____ FT"
      SET LINE 11 TO "GROUND SPEED: _____ KTS"
      SET LINE 12 TO "TRUE AIR SPEED: _____ KTS"
      SET LINE 13 TO "MACH NUMBER: _____"
      SET LINE 14 TO BLANKS
      SET LINE 15 TO "TIME: _____ HRS; _____ MIN; _____ SEC"
      SET SAD_1_RE_INIT_FLAG TO 0
    ENDIF
    READ APPROPRIATE INFORMATION FROM OWN A/C BUFFER
    CONVERT TO ALPHANUMERICS
    UPDATE PROPER LINES
  ENDIF
END

```

Figure 6-57



NADC-79161-40

MODULE NAME: SAD 2 UPDATE

DATA REQUIRED

CONFIGURATION VECTOR(FROM NODE 90)  
-BINARY VECTOR INDICATING CONDITION OF ALL NODES  
-"1" IF WORKING, "0" IF FAILED

DATA PRODUCED:

SAD\_2 OUTPUT BUFFER UPDATE  
-16 LINES OF ALPHANUMERIC DATA FOR SAD 2  
-ONE LINE IS OUTPUT EVERY MINOR CYCLE  
-EACH LINE IS 64 CHARACTERS LONG

SAD 2 UPDATE

THIS MODULE EXECUTES EVERY MINOR CYCLE

-----  
THE ALPHANUMERICS FOR THIS BUFFER ARE INITIALIZED AS FOLLOWS,  
AND ARE ONLY UPDATED WITH THE WORDS "UP" AND "DOWN"

LINE 1: HARDWARE STATUS  
LINE 2: BLANKS  
LINE 3: COMM (NODE 21)  
LINE 4: MAD (NODE 30)  
LINE 5: NAV (NODE 70)  
LINE 6: DISPLAY (NODE 81)  
LINE 7: KEYSER (NODE 82)  
LINE 8: CONTROLLER (NODE 90)  
LINE 9: BUS 1  
LINE 10: BUS 2  
LINES 11 -16 (TBD)

-----  
SET I TO 1  
DOWHILE I LE 8  
. IF BIT(I) OF CONFIGURATION\_VECTOR EQ 1  
. . THEN  
. . APPEND "UP" TO LINE I+2  
. . ELSE  
. . APPEND "DOWN" TO LINE I+2  
. ENDF  
ENDDO  
END

Figure 6-58

## MODULE NAME: SAD 3 OUTPUT PROCESSING

## DATA REQUIRED

TEXT FILE  
 LAST\_CHAR\_IN  
 - POINTER TO LAST CHARACTER ENTERED IN TEXT FILE  
 LAST\_CHAR\_OUT  
 - POINTER TO LAST CHARACTER OUTPUT TO BUFFER  
 LAST\_CR  
 - NUMBER OF CHAR ENTERED IN OUTPUT BUFFER SINCE LAST CARRIAGE RETURN

## DATA PRODUCED

UPDATES OF BUF\_7 (SAD 2 OUTPUT BUFFER - XMT SUB ADDRESS 7)

## SAD 3 OUTPUT PROCESSING

THIS MODULE EXECUTES EVERY MINOR CYCLE

```

IF SAD_3_TEXT_FLAG EQ 1
. SET ALL WORDS IN BUF_7 TO 0
. SET I TO 1
. SET J TO LAST_CR
. DOWHILE LAST_CHAR_OUT LE LAST_CHAR_IN AND I LE 64
. . SET PTR TO LAST_CHAR_OUT + 1
. . SET CHAR TO TEXT FILE CHAR(PTR)
. . IF CHAR EQ CONTROL CHARACTER
. . . THEN
. . . . CASE CHAR
. . . . . -CASE 1: CARRIAGE RETURN
. . . . . SET LAST_CR TO 0
. . . . . ENTER CHAR IN BUF_7(I)
. . . . . INCREMENT I
. . . . . -CASE 2: BACKSPACE
. . . . . IF LAST_CR NE 0 THEN SET LAST_CR TO LAST_CR - 1
. . . . . ENDIF
. . . . . ENTER CHAR IN BUF_7(I)
. . . . . INCREMENT I
. . . . . -CASE 3: ERASE LINE
. . . . . DOWHILE LAST_CR GT 0
. . . . . . ENTER BACKSPACE IN BUF_7(I)
. . . . . . INCREMENT I
. . . . . . DECREMENT LAST_CR
. . . . . ENDDO
. . . . . ENDCASE
. . . . ELSE
. . . . IF LAST_CR LE 64
. . . . . THEN
. . . . . INCREMENT LAST_CR
. . . . . ELSE
. . . . . SET LAST_CR TO 0
. . . . . ENTER CARRIAGE RETURN IN BUF_7(I)
. . . . . INCREMENT I
. . . . . ENDIF
. . . . ENTER CHAR IN BUF_7(I)
. . . . INCREMENT I
. . . . ENDDO
. . SET LAST_CHAR_OUT TO PTR
. ENDDO
. WRITE BUF_7 TO INTERFACE MODULE
ENDIF
END

```

Figure 6-59

NADC-79161-40

MODULE NAME: DISPLAY INPUT PROCESSING

DATA REQUIRED

INPUT MESSAGE BUFFERS  
-IN\_6 (RCV SUB ADDRESS 6 OF TABLE 6-- - 20 WORDS)  
-IN\_7 (32 WDS)  
-IN\_8 (32 WDS)  
-IN\_9 (32 WDS)  
-IN\_10 (32 WDS)  
-IN\_11 (32 WDS)  
-IN\_12 (3 WDS)  
-IN\_16 (19 WDS)  
-IN\_17 (21 WDS)  
-IN\_18 (22 WDS)  
-IN\_19 (24 WDS)  
-IN\_20 (24 WDS)  
-IN\_21 (24 WDS)  
-IN\_22 (32 WDS)  
-IN\_23 (20 WDS)

DATA PRODUCED

UPDATES OF  
-FIX FILE  
-TRACK FILE  
-SONOBOUY FILE  
-TEXT FILE

DISPLAY INPUT PROCESSING

BLOCK 1- MAD INPUT (BUFFERS 6 THRU 12)

-----  
THIS BLOCK WILL ADD NEW TEXT TO THE SAD\_3 TEXT BUFFER.  
IMPLEMENTATION IS DEPENDENT ON THE MODIFICATION OF THE  
MADTACS PROGRAM.  
-----

BLOCK 2- NAV INPUTS (BUFFERS 16 - 23)

BLOCK 2A - PROCESS IN\_16  
COPY DYNAMIC DATA FROM IN\_16 INTO OWN A/C PARAMETER BUFFER

BLOCK 2B - PROCESS IN\_17  
COPY DYNAMIC DATA FROM IN\_17 TO OWN A/C PARAMETER BUFFER

BLOCK 2C - PROCESS IN\_18  
COPY DYNAMIC DATA FROM IN\_18 INTO OWN A/C PARAMETER BUFFER

BLOCK 2D - PROCESS IN\_19, IN\_20, IN\_21 (FIX FILE)

(IT IS ASSUMED THAT THE THREE BUFFERS ARE CONSECUTIVE IN MEMORY)

SET I TO 1  
DOWHILE I LE 6 (6 FIX ENTRIES FILL THREE BUFFERS)  
 . IF DISPLAY CUE OF I<sup>TH</sup> FIX NE 0  
 . . CONVERT U/V CO-ORDINATES OF FIX TO X,Y DISPLAY CO-ORD  
 . . APPEND X,Y DISPLAY CO-ORD TO FIX DATA  
 . . APPEND FIX DATA TO FIX FILE  
 . . INCREMENT NUM\_FIXES  
 . ENDOF  
 . INCREMENT I  
ENDDO

Figure 6-60a

```

BLOCK 2E - PROCESS IN_22 (TRACK FILE)
SET I TO 1
DOWHILE I LE 2
. SET J TO 1
. DOWHILE J LE NJM_TRACKS
. . IF (TRACK ID OF JTH ENTRY IN TRACK FILE) EQ (TRACK ID OF ITH
. . . ENTRY
. . . IN BUFFER IN_22)
. . . THEN
. . . COPY ITH ENTRY FROM IN_22 INTO JTH ENTRY IN TRACK FILE
. . . ENDOF
. . INCREMENT J
. ENDDO
. INCREMENT I
ENDDO

BLOCK 2F - PROCESS IN_23 (SONOBOUY FILE)
IF DISPLAY CUE OF ENTRY IN BUFFER IN_23 NE 0
. CONVERT U/V CO-ORDINATES OF ENTRY TO X,Y DISPLAY CO-ORD
. SET I TO 1
. SET FIND_FLAG TO 0
. DOWHILE I LE NUM_SONOBOUYS AND FIND_FLAG EQ 0
. . IF ((ID OF ENTRY I IN SONOBOUY FILE) EQ (ID OF ENTRY IN BUFFER
. . . IN_23))
. . . SET FIND_FLAG TO 1
. . . APPEND X,Y CO-ORDINATES TO SONOBOUY DATA FROM IN_23
. . . REPLACE APPROPRIATE ENTRY IN SONOBOUY FILE WITH NEW DATA
. . . SET NEW_SONO_FLAG(I) TO 1
. . . ENDOF
. . INCREMENT I
. ENDDO
. IF FIND_FLAG EQ 0
. . APPEND X,Y DISPLAY CO-ORD TO DATA FROM IN_23
. . APPEND NEW DATA TO SONOBOUY FILE
. . INCREMENT NUM_SONOBOUYS
. . APPEND A "1" TO NEW_SONO_FLAG VECTOR
. . ENDOF
ENDIF

BLOCK 3 - EMPTY BUFFERS FOR NEW DATA
SET ALL BUFFER WORDS, IN ALL BUFFERS, TO 0
(0 IS NOT ALLOWED AS A TRACK ID)

END

```

Figure 6-60b



NADC-79161-40

MODULE NAME: DISPLAY SELF TEST

DATA REQUIRED

RESULTS OF SELF TEST PROGRAM  
2 STATUS WORDS  
OLD SEQUENCE WORD (SEQUENCE WORD LAST TRANSMITTED)  
LAST RETURN WORD INPUT FROM NODE 90

DATA PRODUCED

8 WORD SELF TEST MESSAGE (BUF\_29)  
-NODE ID, MAJOR CYCLE NO, MINOR CYCLE NO  
-STATUS WORD 1  
-STATUS WORD 2  
-CONFIGURATION IDENT  
-RESULTS OF SELF TEST PROGRAM  
-OLD SEQUENCE WORD  
-RETURN WORD  
-NEW SEQUENCE WORD

DISPLAY SELF TEST

THIS MODULE EXECUTES EVERY MINOR CYCLE

SET WORD 5 OF BUF\_29 TO RESULT OF SELF TEST PROGRAM  
IF WORD 5 INDICATES A FAILURE  
. THEN  
. IF FAILURE RECOVERABLE  
. . THEN  
. . . CALL APPROPRIATE RECOVERY ROUTINE  
. . . SET STATUS WORDS AND CONFIGURATION ID TO INDICATE NEW  
. . . CONFIGURATION  
. . ELSE  
. . . ATTEMPT TO ACTIVATE SHUTDOWN MECHANISM  
. . ENDIF  
. ELSE  
. IF NEW MAJOR CYCLE  
. . INCREMENT MAJOR CYCLE NO.  
. . SET MINOR CYCLE NO. TO 0  
. . ENDOF  
. SET WORD 1 OF BUF\_29 TO NODE ID, MAJOR CYCLE NO, MINOR CYCLE NO  
. INCREMENT MINOR CYCLE NO.  
. COPY STATUS WORD 1 INTO WORD 2 OF BUF\_29  
. COPY STATUS WORD 2 INTO WORD 3 OF BUF\_29  
. SET WORD 4 OF BUF\_29 TO CONFIGURATION ID  
. SET WORD 6 OF BUF\_29 TO LAST WORD 8 TRANSMITTED FROM BUF\_29  
. SET WORD 7 OF BUF\_29 TO LAST WORD 8 RECEIVED FROM NODE 90  
. SET WORD 8 OF BUF\_29 TO A NEW SEQUENCE WORD  
. WRITE BUF\_29 TO I.M.  
ENDIF  
END

Figure 6-61. Node 81 Test and Recovery Function

### 6.6.3 Node 30

#### 6.6.3.1 General Description

Node 30 performs two major functions. It performs MAD processing, and it acts as a monitor/backup for Node 90. If Node 90 (the bus controller) fails, Node 30 will initiate control of the bus.

#### 6.6.3.2 I/O Description

Node 30 is relatively self-contained. It receives MAD sensor data from Node 115 and navigation and contact information from Node 70, processes the data, and alerts Node 70 (the navigation subsystem) if a detection is made. This node accepts control commands from Node 82 and sends display data to Node 82, if required. Node 30 input and output messages for the implementation subset are shown in Tables 6-13 and 6-14, respectively.

For the purpose of this implementation subset, the use of Node 30 synchronous output messages to Node 82 was considered sufficient to transmit to the display those statistical data that are produced as output by the existing MAD program described in the previous paragraph. Data rates and bus loading will therefore be maintained, but meaningful data will still be passed to the display.

Node 30 also sends a self-test message to the bus controller every minor cycle and likewise receives a self-test message from the controller.

#### 6.6.3.3 Software Modules Descriptions

6.6.3.3.1 Executive Control Processing. Assume that the SDEX-M (Reference 22) executive will be implemented in Node 30, since the node is an AN/AYK-14 computer. This executive functions when an application process makes an executive service request (ESR).

It will also be necessary to have the bus controller program resident in Node 30 in case Node 90 fails.

TABLE 6-13. NODE 30 INPUT MESSAGES

	SRC NODE	RCV SUB ADR	WORDS PER SECOND	PERIOD IN MSEC	CYC CODE	WORD COUNT
***						
*	70	1	300	50	0	15
*	70	2	420	50	0	21
*	82	30	3	0		3
*	90	29	160	50	0	8
*	90	30		0		3
*	21	3	48	50	0	3
*	21	22	60	50	0	3
*	21	10	120	50	2	24
						1

TABLE 6-14. NODE 30 OUTPUT MESSAGES

	DEST. NODE	XMT SUB ADR	WORDS PER SECOND	PERIOD IN MSEC	CYC CODE	WORD COUNT
***						
*	70	30	3	0		3
*	82	1	400	40	0	20
*	82	2	3600	160	0	32
*		3			0	32
*		4			0	32
*		5			0	32
*		6			0	32
*		7			0	32
*	82	30	3	0		3
*	90	29	160	50	0	8
*	21	30	3	0		3
						1

6.6.3.3.2 Operational Modules. These modules perform MAD processing, as a result of receiving input command from Node 82.

- a. Input Command Processing — Input Command Processing is a module invoked when an asynchronous bus message is received by Node 30. This module interprets the first message word as a command word. The remaining message words contain any necessary data for the execution of the command. Figures 6-62 through 6-67 present functional descriptions of the Input Command Processing module and its associated subroutines.
- b. MAD Processing — The MAD Processing module detects and localizes a target based on the outputs of simulated magnetometer signals (one frequency value every 50 milliseconds). This program was originally written for the CP-IO machine in the SPL/I language by Code 503 personnel. The SPL/I compiler for the AN/AYK-14 is now available and allows this program to be used to exercise the AN/AYK-14 in this simulation problem. Minor modifications will have to be made in the I/O routines, but these should have minimum cost impact. Figures 6-68 through 6-72 present a structured description of the program, while Appendix A<sup>1</sup> is a listing of the compilation of the program.

6.6.3.3.3 Test and Reconfiguration IV. Node 30 monitors Node 90 (Bus Control) and acts as backup for it in case of failure. This node, like every node, performs self-test program every minor cycle.

- a. Controller Monitor (see Figure 6-73) — Node 30 is designated as the backup controller for Node 90. As such, it receives and verifies the Node 90 self-test messages produced by the periodic execution of the Node 90 self-test program. If the Node 90 self-test message indicates a failure, Node 30 will verify the self-test message by checking the message Node 30 received with the same message as received by Node 70. If the Node 70 message also indicates a failure, a discrete interrupt line is activated which will remove power from Node 90, thus isolating it from the bus. Node 30 will then take over test and monitor functions for the system through execution of the reconfiguration program.
- b. Reconfiguration (see Figure 6-74) — This program loads in the programs normally executed by Node 70 and schedules them as Node 30 tasks; this procedure allows recovery from a Node 90 failure.
- c. Self-Test (Figure 6-75) — During every minor cycle, Node 30 will execute a self-test program and format an appropriate message to the bus controller indicating its status. The self-test program is the IFPM program developed for the AN/AYK-14, and the message format is shown in Table 6-7.

<sup>1</sup> Appendix A is a separate volume available from NAVAIRDEVCEEN Code 502.



NADC-79161-40

Figures 6-62 through 6-67. Node 30 Input Processing Function

NADC-79161-40

MODULE NAME: INPUT COMMAND PROCESSING

DATA REQUIRED

MODE 62 ASYNCHRONOUS MSG  
\_COMMAND WORD  
\_DATA WORD 1  
\_DATA WORD 2

DATA PRODUCED

NONE

INPUT COMMAND PROCESSING

THIS MODULE INTERPRETS THE COMMAND AND CALLS THE APPROPRIATE  
LOCAL SUBROUTINE  
CASE COMMAND WORD  
  . SHAD RE-INIT\$  
  . SHAD DISPLAY\$  
  . \$ACTIVATE FRP\$  
  . SHAD MARK\$  
  . SHAD DISABLE\$  
ENDCASE  
END

Figure 6-62

SHAD RE-INIT

THIS SUBROUTINE RESETS THE INITIAL VALUES OF THE SHAD SUBPROGRAM  
TO PRESET VALUES  
END

Figure 6-63

NADC-79161-40

**MAD DISPLAY**

THIS SUBROUTINE SETS THE FORMAT OPTIONS FOR THE  
\*MAD PROCESSING\* MODULE ACCORDING TO THE VALUE OF DATA WORD 1,  
AND ACTIVATES THE \*MAD PROCESSING\* TASK

END

Figure 6-64

**ACTIVATE FRP**

THIS SUBROUTINE SETS THE FLAG INDICATING THAT FEATURE RECOGNITION  
PROCESSING IS TO BE PERFORMED BY THE \*MAD PROCESSING\* MODULE

SET FRP\_OUTPUT\_FLAG TO 1

END

Figure 6-65

**MAD MARK**

THIS MODULE SENDS AN "ENTER FIX" COMMAND TO  
MODE 70 (NAV), WITH THE FIX ID INDICATED IN DATA WORD 1

END

Figure 6-66

**MAD DISABLE**

THIS SUBROUTINE TERMINATES THE \*MAD PROCESSING\* MODULE

END

Figure 6-67

Figures 6-68 through 6-72. Node 30 MAD Processing Function



MODULE NAME: (MAD) MAGNETIC ANOMALY DETECTION

DATA REQUIRED

INPUT SIGNAL DATA  
SAMPLE WINDOW FOR EACH CHANNEL  
PREPARE DETECT FLAG  
LOCALIZATION FLAG  
SEQUENCE  
TABLE FOR EACH SEQUENCE  
NUMBER OF SEQUENCES  
EXPECTED CHANNEL OF EACH SEQUENCE  
CHANNEL PROCESSOR FLAGS  
TIME LIMIT CLOCK FOR EACH CHANNEL  
LOCK OUT TIMER FOR EACH CHANNEL

DATA PRODUCED

SAMPLE WINDOW FOR EACH CHANNEL  
CHANNEL PROCESSOR FLAGS  
PREPARE DETECT FLAG  
SLANT RANGE  
CONFIDENCE LEVEL  
RETROMARK TIME  
LOCALIZATION STAT FLAG  
NUMBER OF SEQUENCES  
LOCALIZATION FLAG  
EXPECTED CHANNEL OF EACH SEQUENCE  
SEQUENCE  
CHANNEL  
TIME LIMIT CLOCK FOR EACH CHANNEL  
LOCK OUT TIMER FOR EACH CHANNEL

Figure 6-68a

```

*****
* PREPROCESSING CYCLE
*
LOWPASS FILTER INPUT SIGNAL DATA
PUT DATA IN SAMPLE WINDOWS OF APPROPRIATE CHANNELS AND SET THE
CORRESPONDING CHANNEL PROCESSOR FLAGS (SAMPLE VECTOR GENERATION)
*
*END OF PREPROCESSING CYCLE
*****
(EXECUTION CYCLE: DETERMINE AND PROCESS THE HIGHEST PRIORITY TASK)
IF PREPARE DETECT IS AWAITING EXECUTION
. THEN
. $DETECTIONS
. *****
. * LIKELIHOOD COMPUTATIONS
. * SEQUENCE PROCESSING
. *****
. RESET PREPARE DETECT FLAG
. ELSE
. IF LOCALIZATION IS AWAITING EXECUTION
. . THEN
. . *****
. . * LOCALIZATION CYCLE
. . *
. . * LOCALIZE USING THE INFORMATION FROM THE SEQUENCE'S TABLE
. . *
. . COMPUTE - SLANT RANGE, CONFIDENCE LEVEL, AND RETROMARK TIME
. . DISPLAY STATISTICS (SET LOCALIZATION STAT FLAG)
. . HAVING CONCLUDED SEQUENCE PROCESSING DECREMENT NUMBER OF
. . SEQUENCES
. . IF THE OTHER SEQUENCE DOES NOT REQUIRE PROCESSING
. . . (THE EXPECTED CHANNEL OF THE SEQUENCES DIFFER)
. . . THEN
. . . RESET LOCALIZATION FLAG
. . . ENDIF
. . SET EXPECTED CHANNEL OF SEQUENCE TO NONE (7)
. . SET SEQUENCE (TABLE POINTER) TO THE OTHER SEQUENCE
. . *
. . *END OF LOCALIZATION CYCLE
. . *****
. . ELSE
. . IF A CHANNEL PROCESSOR IS AWAITING EXECUTION
. . . THEN
. . . $PROCESS CHANNELS FOR THE LOWEST CHANNEL PROCESSOR WAITING
. . . *****
. . . * SIGNAL DESCRIPTION
. . . * PRELIMINARY DETECTION TESTING / EPOCH TRACKING
. . . *****
. . . UPDATE CHANNEL'S TIME LIMIT CLOCK
. . . UPDATE CHANNEL'S LOCK OUT TIMER
. . . RESET CHANNEL PROCESSOR FLAG
. . . ENDIF
. . ENDIF
ENDIF
(WAIT FOR NEXT INPUT SIGNAL DATA)

```

END

Figure 6-68b

MODULE NAME: DETECTION

DATA REQUIRED

CHANNEL  
LIKELIHOOD PRODUCT THRESHOLD FOR EACH CHANNEL  
EXPECTED CHANNEL OF EACH SEQUENCE  
LOCK OUT TIMER FOR EACH CHANNEL  
NUMBER OF SEQUENCES  
SEQUENCE  
LENGTH OF EACH SEQUENCE  
SAVED COPY OF SIGNAL METRICS  
TIME LIMIT CLOCK

DATA PRODUCED

TIMESLICE STATISTICS  
LIKELIHOOD PRODUCT  
DETECTION FLAG FOR CHANNEL  
SEQUENCE  
SEQUENCE TABLE POINTER  
LENGTH OF EACH SEQUENCE  
NUMBER OF SEQUENCES  
LOCK OUT TIMER FOR EACH CHANNEL  
TIME LIMIT CLOCK FOR EACH CHANNEL  
ALERT FLAG  
EPOCH FLAG OF CHANNEL  
EXPECTED CHANNEL OF EACH SEQUENCE  
SEQUENCE TABLE STATISTICS  
LOCALIZATION FLAG  
SAVED COPY OF SIGNAL METRICS

Figure 6-69a

## DETECTION

```

COMPUTE TIMESLICE STATISTICS AND THE LIKELIHOOD PRODUCT
IF LIKELIHOOD PRODUCT EXCEEDS THIS CHANNEL'S LIKELIHOOD PRODUCT
.   THRESHOLD
.   THEN
.   .   (PROCESS SEQUENCE)
.   .   SET DETECTION FLAG OF CHANNEL
.   .   IF CHANNEL WAS NOT EXPECTED BY EITHER SEQUENCE
.   .   .   THEN
.   .   .   .   (TRY TO INITIATE A NEW SEQUENCE)
.   .   .   .   IF LOCKOUT TIMER OF THIS CHANNEL HAS NOT EXPIRED OR THE NUMBER OF
.   .   .   .   .   SEQUENCES EXCEEDS 1
.   .   .   .   .   THEN
.   .   .   .   .   .   EXIT FROM PREPARE FOR DETECTION
.   .   .   .   .   ELSE
.   .   .   .   .   .   (INITIATE NEW SEQUENCE)
.   .   .   .   .   .   INCREMENT NUMBER OF SEQUENCES
.   .   .   .   .   .   SET SEQUENCE TO THE FIRST AVAILABLE TABLE
.   .   .   .   .   .   SET LENGTH OF THIS SEQUENCE TO 0
.   .   .   .   .   .   SAVE CHANNEL INITIATING SEQUENCE
.   .   .   .   .   .   SET THE LOCK OUT TIMERS FOR THIS AND HIGHER CHANNELS
.   .   .   .   .   .   SET THE TIME LIMIT CLOCKS FOR THE NEXT 2 CHANNELS
.   .   .   .   .   .   SAVE AIRCRAFT VELOCITY
.   .   .   .   .   .   ACTIVATE CONFIRMING ALERT (SET ALERT FLAG)
.   .   .   .   .   ENDIF
.   .   .   .   ENDIF
.   .   .   .   DOWHILE CHANNEL IS EXPECTED BY SEQUENCE
.   .   .   .   .   INCREMENT LENGTH OF SEQUENCE
.   .   .   .   .   SET EPOCH FLAG OF CHANNEL TO SEQUENCE
.   .   .   .   .   SET EXPECTED CHANNEL OF SEQUENCE TO THE NEXT CHANNEL
.   .   .   .   .   TRANSFER SAVED SIGNAL METRICS TO SEQUENCE TABLE
.   .   .   .   .   SAVE TIME OF CONFIRMATION AND LIKELIHOOD PRODUCT
.   .   .   .   .   SET SEQUENCE TO 2
.   .   .   .   ENDDO
.   .   ELSE
.   .   .   SAVE CURRENT SIGNAL METRICS
.   .   .   IF TIME LIMIT CLOCK OF THIS CHANNEL HAS EXPIRED WHILE THE CHANNEL
.   .   .   .   WAS EXPECTED BY ONE OF THE SEQUENCES
.   .   .   .   THEN
.   .   .   .   .   SET LOCALIZATION FLAG FOR PROCESSING OF TARGET INFORMATION ON THE
.   .   .   .   .   .   NEXT CYCLE AS NO MORE CHANNEL INFORMATION IS ADMISSIBLE
.   .   .   .   ENDIF
.   .   ENDIF
ENDIF
END

```

Figure 6-69b



MODULE NAME: PROCESS CHANNEL

DATA REQUIRED

CHANNEL  
SAMPLE WINDOW  
AVERAGE POWER  
SIGNAL POWER HISTORY  
GAMMA HISTORY  
EPOCH FLAG FOR CHANNEL  
EPOCH HISTORY  
TIME LIMIT CLOCK FOR THIS AND NEXT CHANNEL  
EXPECTED CHANNEL OF EACH SEQUENCE  
DETECTION FLAG FOR CHANNEL  
SAVED AVERAGE POWER LEVEL

DATA PRODUCED

SAMPLE VECTOR  
SIGNAL METRICS  
-INPUT SIGNAL POWER  
-ANDERSON COEFFICIENTS  
-SIGNAL POWER ESTIMATE  
-GAMMA  
-NOISE POWER  
AVERAGE POWER  
SIGNAL EVENT THRESHOLD  
SIGNAL POWER HISTORY  
GAMMA HISTORY  
EPOCH  
EPOCH HISTORY  
LOCALIZATION FLAG  
EPOCH FLAG FOR CHANNEL  
SAVED AVERAGE POWER LEVEL  
PREPARE DETECT FLAG  
SAVED COPY OF SIGNAL METRICS  
DETECTION FLAG FOR CHANNEL

Figure 6-70a

# SPROCESS CHANNELS

NADC-79161-40

```

GENERATE CHANNEL'S NORMALIZED SAMPLE VECTOR FROM SAMPLE WINDOW BY
  REMOVING RAMP AND DC BIAS
CHARACTERIZE SIGNAL BY CALCULATING: INPUT SIGNAL POWER, ANDERSON
  COEFFICIENTS, SIGNAL POWER ESTIMATE, GAMMA, NOISE POWER, AVERAGE
  POWER, SIGNAL EVENT THRESHOLD
UPDATE SIGNAL POWER AND GAMMA HISTORIES
COMPUTE EPOCH (DETERMINATOR) AND UPDATE ITS HISTORY
IF AN EVENT HAS BEEN CONFIRMED ON THIS CHANNEL (EPOCH FLAG OF THIS
  CHANNEL IS NOT 0)
  THEN
    IF A MINIMUM IN EPOCH HAS OCCURRED
      THEN
        SET SEQUENCE TO EPOCH FLAG OF CHANNEL
        SAVE DATA FOR EPOCH DETERMINATOR
        IF THIS IS CHANNEL 6 OR SEQUENCE LENGTH IS 3 OR CHANNEL+1 IS
          EXPECTED AND ITS TIME LIMIT CLOCK HAS EXPIRED
            THEN
              SET LOCALIZATION FLAG
            ENDIF
          RESET EPOCH FLAG OF CHANNEL TO 0
        ENDIF
      ELSE
        SET SEQUENCE TO FIRST TABLE EXPECTING THIS CHANNEL (OR 0)
        IF AN EVENT OCCURRED (SIGNAL POWER EXCEEDS EVENTTHRESHOLD)
          THEN
            SAVE CURRENT AVERAGE POWER LEVEL AS BASIS FOR NEXT EVENT AS
              REQUIRED
            IF A MAXIMUM IN GAMMA HAS OCCURRED
              THEN
                SET PREPARE DETECT FLAG
              ELSE
                SAVE CURRENT SIGNAL METRICS
                IF TIME LIMIT CLOCK OF THIS CHANNEL HAS EXPIRED WHILE THE
                  CHANNEL WAS EXPECTED BY ONE OF THE SEQUENCES
                    THEN
                      SET LOCALIZATION FLAG FOR PROCESSING OF TARGET INFORMATION
                        ON THE NEXT CYCLE AS NO MORE CHANNEL INFORMATION IS
                          ADMISSIBLE
                    ENDIF
                ENDIF
              ELSE
                IF THE DETECTION FLAG OF CHANNEL IS SET
                  THEN
                    RESTORE AVERAGE POWER LEVEL TO THE VALUE SAVED WHEN THE EVENT
                      OCCURRED
                    RESET DETECTION FLAG OF CHANNEL
                  ENDIF
                IF TIME LIMIT CLOCK OF THIS CHANNEL HAS EXPIRED WHILE THE CHANNEL
                  WAS EXPECTED BY ONE OF THE SEQUENCES
                    THEN
                      SET LOCALIZATION FLAG FOR PROCESSING OF TARGET INFORMATION ON
                        THE NEXT CYCLE AS NO MORE CHANNEL INFORMATION IS ADMISSIBLE
                    ENDIF
                ENDIF
              ENDIF
            ENDIF
          ENDIF
        ENDIF
      ENDIF
    ENDIF
  ENDIF
ENDIF

```

END

Figure 6-70b

## MODULE NAME: CONTROL PROGRAM FOR MAD

## DATA REQUIRED

ALERT FLAG  
 LOCALIZATION STAT FLAG  
 MADBOX OPERATION FLAG  
 MISC. RUN TIME OPTIONS

## DATA PRODUCED

ALERT FLAG  
 LOCALIZATION STAT FLAG

## CONTROL PROGRAM FOR MAD

SET INITIAL CONDITIONS FOR PROGRAM ACCORDING TO RUN TIME SETTINGS  
 IF MADBOX OPERATION IS REQUESTED

```

. THEN
.   REGISTER 3MADLOOPING PROGRAMS
.   TURN ON MAD ISC BOX (WHICH WILL INVOKE MADLOOPING PROGRAM AT 1/20TH
.     SECOND INTERVALS)
. ELSE
.   (OPERATE OFF OF MAG-TAPE DATA)
.   DISCARD FIRST RECORD OF DATA (OPTIONAL)
.   DOWHILE (TRUE)
.     . READ A RECORD OF DATA - 126 SAMPLES (OPTIONAL THE FIRST TIME)
.     . OPTIONALLY DUMP DATA TO PRINTER
.     . SET COUNTER TO 1
.     . DOWHILE COUNTER IS LESS THAN 126
.     .   EXECUTE MAD WITH SCALED SAMPLE (COUNTER)
.     .   . OPTIONALLY DUMP INTERMEDIATE RESULTS
.     .   . IF ALERT FLAG IS SET
.     .   .   THEN
.     .   .     RESET ALERT FLAG
.     .   .     SIGNAL THE ALERT ON THE LINE PRINTER
.     .   .   ENDIF
.     .   . IF LOCALIZATION STAT FLAG IS SET
.     .   .   THEN
.     .   .     RESET LOCALIZATION STAT FLAG
.     .   .     OUTPUT STATISTICS ON THE HAZELTINE OR THE LINE PRINTER
.     .   .     PRINT STATISTICS
.     .   .   ENDIF
.     .   INCREMENT COUNTER
.     . ENDDO
.     . OPTIONALLY DUMP CHANNEL HISTORY STATISTICS
.     . OPTIONALLY STOP EXECUTION FOR OPERATOR INTERVENTION
.   ENDDO
. ENDIF
END
```

Figure 6-71

MADLOOPING PROGRAM FOR REAL-TIME OPERATION WITH SUB-SIMULATOR

```

(INVOKED AS A SELECTED EVENT TASK ON RECEIVING AN INTERRUPT AND INPUT
 SIGNAL FROM THE ISC)
MASK AND SCALE INPUT SIGNAL
IF MAG DUMP FLAG IS SET
. THEN
. DUMP SCALED SIGNALS ONTO TAPE (SAME FORMAT AS USED FOR TAPE
. OPERATION)
. ELSE
. EXECUTE MAD WITH SCALED SIGNAL
. IF ALERT FLAG IS SET
. . THEN
. . RESET ALERT FLAG
. . SIGNAL THE ALERT ON THE LINE PRINTER
. ENDIF
. IF LOCALIZATION STAT FLAG IS SET
. . THEN
. . RESET LOCALIZATION STAT FLAG
. . OUTPUT STATISTICS ON THE HAZELTINE OR THE LINE PRINTER
. . PRINT STATISTICS
. ENDIF
ENDIF
END

```

Figure 6-72



MODULE NAME: CONTROLLER MONITOR

DATA REQUIRED

NODE 90 OUTPUT SELF TEST MESSAGE  
 NODE 79 REPEAT OF NODE 90 OUTPUT MESSAGE

DATA PRODUCED

NONE

CONTROLLER MONITOR

THIS MODULE IS ACTIVATED BY A  
 CYCLE OR BY AN INTERRUPT WHICH OCCURS WHEN A CERTAIN PRESET AMOUNT  
 OF TIME OCCURS WITHOUT A NEW MINOR CYCLE COMMAND BEING  
 RECEIVED  
 VERIFY NODE ID, MAJOR CYCLE NO, MINOR CYCLE NO. FROM WORD 1  
 OF RECEIVED MESSAGE  
 VERIFY STATUS WORDS OK (WORDS 2 AND 3 OF MESSAGE)  
 SAVE CONFIGURATION ID  
 VERIFY IFPM RESULTS OK  
 VERIFY THAT LAST WORD 8 OF NODE 30 SELF TEST MESSAGE SENT  
 EQUALS WORD 7 OF THIS MESSAGE  
 VERIFY THAT LAST WORD 8 OF NODE 90 SELF TEST MESSAGE  
 EQUALS WORD 6 OF THIS MESSAGE  
 SAVE WORD 8 OF THIS MESSAGE  
 IF ANY VERIFY FAILS  
 . THEN  
 . ACTIVATE BUS CONTROL FOR BUS 2  
 . READ LAST NODE 90 SELF TEST MESSAGE FROM NODE 79  
 . IF MESSAGES DO NOT AGREE  
 . . THEN  
 . . SIGNAL OPERATOR 'SYSTEM FAILURE NOT ISOLATED'  
 . . ELSE  
 . . SIGNAL OPERATOR 'NODE 90 FAILURE; RECONFIGURATION ATTEMPTED'  
 . . ACTIVATE DISCRETE NODE 90 SHUTDOWN INTERRUPT (PREFERABLY A POWER  
 . . SHUTDOWN)  
 . . CALL 'RECONFIGURATION'  
 . ENDIF  
 ENOIF  
 END

Figure 6-73. Node 30 Controller Monitor Function

MODULE NAME: RECONFIGURATION

DATA REQUIRED

RECONFIGURATION PROGRAMS FROM DISK THROUGH NODE 21

DATA PRODUCED

SIGNAL TO OPERATOR 'RECONFIGURATION COMPLETE'

RECONFIGURATION

```
SUSPEND NEW MINOR CYCLE MESSAGES
SET LAST TO 0
DOWHILE LAST EQ 0
. WRITE 'READ BLOCK' COMMAND TO MASS MEMORY (NODE 21)
. LOOP UNTIL BLOCK IS RECEIVED
. TRANSFER BLOCK TO PROPER MEMORY LOCATION
. IF ALL PROGRAMS TRANSFERRED
. . THEN
. . SET LAST TO 1
. . ENDOIF
ENDDO
RESET MINOR CYCLE TIME
ACTIVATE *COMMAND PROCESSING* MODULE
ACTIVATE *CONFIGURATION MONITOR* MODULE
ACTIVATE *STATISTICS* MODULE
ACTIVATE *SELF TEST* MODULE
ACTIVATE *MAD PROCESSING* MODULE
SIGNAL OPERATOR THAT RECONFIGURATION IS COMPLETE
END
```

Figure 6-74. Node 30 Reconfiguration Function

NADC-79161-40

MODULE NAME: MAD SELF TEST

DATA REQUIRED

RESULTS OF SELF TEST PROGRAM  
2 STATUS WORDS  
OLD SEQUENCE WORD (SEQUENCE WORD LAST TRANSMITTED)  
LAST RETURN WORD INPUT FROM NODE 90

DATA PRODUCED

8 WORD SELF TEST MESSAGE (BUF\_29)  
-NODE ID, MAJOR CYCLE NO, MINOR CYCLE NO  
-STATUS WORD 1  
-STATUS WORD 2  
-CONFIGURATION IDENT  
-RESULTS OF SELF TEST PROGRAM  
-OLD SEQUENCE WORD  
-RETURN WORD  
-NEW SEQUENCE WORD

MAD SELF TEST

THIS MODULE EXECUTES EVERY MINOR CYCLE

SET WORD 5 OF BUF\_29 TO RESULT OF SELF TEST PROGRAM  
IF WORD 5 INDICATES A FAILURE  
• THEN  
• IF FAILURE RECOVERABLE  
• • THEN  
• • CALL APPROPRIATE RECOVERY ROUTINE  
• • SET STATUS WORDS AND CONFIGURATION ID TO INDICATE NEW  
• • CONFIGURATION  
• • ELSE  
• • ATTEMPT TO ACTIVATE SHUTDOWN MECHANISM  
• ENDOF  
• ELSE  
• IF NEW MAJOR CYCLE  
• • INCREMENT MAJOR CYCLE NO.  
• • SET MINOR CYCLE NO. TO 0  
• ENDOF  
• SET WORD 1 OF BUF\_29 TO NODE ID, MAJOR CYCLE NO, MINOR CYCLE NO  
• INCREMENT MINOR CYCLE NO.  
• COPY STATUS WORD 1 INTO WORD 2 OF BUF\_29  
• COPY STATUS WORD 2 INTO WORD 3 OF BUF\_29  
• SET WORD 4 OF BUF\_29 TO CONFIGURATION ID  
• SET WORD 6 OF BUF\_29 TO LAST WORD 8 TRANSMITTED FROM BUF\_29  
• SET WORD 7 OF BUF\_29 TO LAST WORD 8 RECEIVED FROM NODE 90  
• SET WORD 8 OF BUF\_29 TO A NEW SEQUENCE WORD  
• WRITE BUF\_29 TO I.M.  
• ENDOF  
END

Figure 6-75. Node 30 Self-Test Function

#### 6.6.4 Node 90

##### 6.6.4.1 General Description

Node 90 performs two major functions. It controls the bus by means of the bus controller program described in the DAIS Bus Control Specification, and it monitors the condition of the system by means of the self-test messages it receives every minor cycle.

##### 6.6.4.2 I/O Description

In addition to sending out the I/O inherent in the bus controller, Node 90 sends out a self-test message to Node 30 and Node 70 every minor cycle. It also receives self-test messages from the other nodes every minor cycle. These messages are defined in Tables 6-15 and 6-16.

##### 6.6.4.3 Software Module Description

6.6.4.3.1 Executive Control Processing. Since Node 90 is an AN/AYK-14 computer, assume that the SDEX-M executive and DAIS bus controller software will be implemented.

6.6.4.3.2 Operational Modules. The DAIS bus controller is the only operational module executed by Node 90.

6.6.4.3.3 Test and Reconfiguration. Node 90 processes all incoming self-test messages; determines if a node is faulty; and, if so, institutes reconfiguration efforts. Figures 6-76 through 6-78 describe this process. Node 90 also performs a self-test every minor cycle and sends the results to both Node 30 and Node 70, as described in Figure 6-78.



NADC-79161-40

TABLE 6-15. NODE 90 INPUT MESSAGES

	SRC NODE	RCV SUB ADR	WORDS PER SECOND	PERIOD IN MSEC	CYC CODE	WORD COUNT
***						
*	21	2	160	50	0	8
*	30	4	160	50	0	8
*	70	5	160	50	0	8
*	81	8	160	50	0	8
*	82	9	160	50	0	8
						1

TABLE 6-16. NODE 90 OUTPUT MESSAGES

	DEST. NODE	XMT SUB ADR	WORDS PER SECOND	PERIOD IN MSEC	CYC CODE	WORD COUNT
***						
*	21	1	160	50	0	8
*	21	30		0		7
*	30	2	160	50	0	8
						1

NADC-79161-40

TEST CASE	TEST CASE	TEST CASE	TEST CASE	TEST CASE	TEST CASE
1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30

Figures 6-76 through 6-78. Node 90 Test and Reconfiguration Function

TEST CASE	TEST CASE	TEST CASE	TEST CASE	TEST CASE	TEST CASE
1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30

NADC-79161-40

MODULE NAME: CONFIGURATION MONITOR

DATA REQUIRED

SELF TEST MESSAGES FROM ALL OTHER NODES  
FIX FILE FROM NODE 82 OR NODE 70  
TRACK FILE FROM NODE 82 OR NODE 70  
SONOBUOY FILE FROM NODE 82 OR NODE 70  
RECOVERY PROGRAMS FROM MASS MEMORY  
RECOVERY DATA AND TASK PARAMETER PACKETS FROM MASS MEMORY

DATA PRODUCED

NONE

CONFIGURATION MONITOR

THIS MODULE PROCESSES THE SELF TEST MESSAGE FROM EACH NODE  
AND DETERMINES WHETHER THAT NODE MUST BE SHUT DOWN AND ISOLATED  
FROM THE SYSTEM. IN THIS EVENT, THE NODE IS ISOLATED AND PRESET  
RECOVERY PROGRAMS AND DATA ARE LOADED FROM MASS MEMORY.  
DYNAMIC DATA IS LOADED FROM NODE 82 OR NODE 70.  
WHERE IT IS REDUNDANTLY STORED.

```
$PROCESS SELF TEST MESSAGES$  
$CALCULATE CONFIGURATION IDS$  
IF RECOVERY MUST BE ATTEMPTED  
  . IF NODE 81 HAS NOT FAILED  
  . . THEN  
  . . SEND ALERT TO OPERATOR  
  . . ELSE  
  . . SEND ALERT TO CGU  
  . ENDOF  
  . INHIBIT ALL MESSAGES TO AND FROM FAILED NODE  
  . CASENTRY FAILED NODE  
  . $NODE 70 RECOVER$  
  . $NODE 21 RECOVER$  
  . $NODE 30 RECOVER$  
  . $NODE 81 RECOVER$  
  . $NODE 82 RECOVER$  
  . $BUS RECOVER$  
ENDCASE  
ACTIVATE COLD START INITIALIZATION  
ENDIF  
END
```

Figure 6-76

## PROCESS SELF TEST MESSAGES

THIS MODULE CHECKS THE CONTENTS OF THE SELF TEST MESSAGES INPUT FROM EACH NODE EVERY MINOR CYCLE. IT RETURNS A BINARY FLAG VECTOR INDICATING WHICH NODES HAVE FAILED.

```

SET NODE_INDEX TO 1
DOWHILE NODE_INDEX LE NUMBER_OF_NODES
. SEPARATE FIRST WORD OF SELF TEST MESSAGE FROM NODE (NODE_INDEX)
. INTO NODE_ID, MAJOR_CYCLE_NO, AND MINOR_CYCLE_NO
. IF ALL CORRECT
. . THEN
. . . CHECK THAT WORD 4 EQ CONFIGURATION_ID
. . . IF CORRECT
. . . . THEN
. . . . . CHECK THAT IFPM RESULTS ARE NORMAL
. . . . . IF CORRECT
. . . . . . THEN
. . . . . . . CHECK THAT WORD 6 EQ LAST WORD 8 TRANSMITTED FROM NODE
. . . . . . . IF CORRECT
. . . . . . . . THEN
. . . . . . . . . CHECK THAT WORD 7 EQ LAST WORD 8 TRANSMITTED TO NODE
. . . . . . . . . IF CORRECT
. . . . . . . . . . THEN
. . . . . . . . . . . SET FAILED_NODE_FLAG(NODE_INDEX) TO 0
. . . . . . . . . . . ELSE
. . . . . . . . . . . . SAVE SELF TEST MESSAGE IN PRESET MEMORY LOCATIONS
. . . . . . . . . . . . SET FAILED_NODE_FLAG(NODE_INDEX) TO 1
. . . . . . . . . . . ENDOIF
. . . . . . . . . . ELSE
. . . . . . . . . . . SET FAILED_NODE_FLAG(NODE_INDEX) TO 1
. . . . . . . . . ENDOIF
. . . . . . ELSE
. . . . . . . SET FAILED_NODE_FLAG(NODE_INDEX) TO 1
. . . . . ENDOIF
. . . ELSE
. . . . SET FAILED_NODE_FLAG(NODE_INDEX) TO 1
. . ENDOIF
. . ELSE
. . . SET FAILED_NODE_FLAG(NODE_INDEX) TO 1
. ENDOIF
. IF FAILED_NODE_FLAG(NODE_INDEX) EQ 1
. . SAVE SELF TEST MESSAGE IN PRESET MEMORY LOCATIONS
. ENDOIF
ENDDO
END

```

Figure 6-77



NADC-79161-40

MODULE NAME: CONTROLLER TEST MESSAGE

DATA REQUIRED

RESULTS OF SELF TEST PROGRAM  
2 STATUS WORDS  
OLD SEQUENCE WORD (SEQUENCE WORD LAST TRANSMITTED)  
LAST RETURN WORD INPUT FROM NODES 30

DATA PRODUCED

8 WORD SELF TEST MESSAGE (BUF\_29)  
-NODE ID, MAJOR CYCLE NO, MINOR CYCLE NO  
-STATUS WORD 1  
-STATUS WORD 2  
-CONFIGURATION IDENT  
-RESULTS OF SELF TEST PROGRAM  
-OLD SEQUENCE WORD  
-RETURN WORD  
-NEW SEQUENCE WORD

CONTROLLER TEST MESSAGE

THIS MODULE EXECUTES EVERY MINOR CYCLE

SET WORD 5 OF BUF\_29 TO RESULT OF SELF TEST PROGRAM  
IF WORD 5 INDICATES A FAILURE

```
. THEN
. IF FAILURE RECOVERABLE
. . THEN
. . . CALL APPROPRIATE RECOVERY ROUTINE
. . . SET STATUS WORDS AND CONFIGURATION ID TO INDICATE NEW
. . . CONFIGURATION
. . ELSE
. . . ATTEMPT TO ACTIVATE SHUTDOWN MECHANISM
. . ENDF
. ELSE
. IF NEW MAJOR CYCLE
. . INCREMENT MAJOR CYCLE NO.
. . SET MINOR CYCLE NO. TO 0
. . ENDF
. SET WORD 1 OF BUF_29 TO NODE ID, MAJOR CYCLE NO, MINOR CYCLE NO
. INCREMENT MINOR CYCLE NO.
. COPY STATUS WORD 1 INTO WORD 2 OF BUF_29
. COPY STATUS WORD 2 INTO WORD 3 OF BUF_29
. SET WORD 4 OF BUF_29 TO CONFIGURATION ID
. SET WORD 6 OF BUF_29 TO LAST WORD 8 TRANSMITTED FROM BUF_29
. SET WORD 7 OF BUF_29 TO LAST WORD 8 RECEIVED FROM NODE 30
. SET WORD 8 OF BUF_29 TO A NEW SEQUENCE WORD
. WRITE BUF_29 TO NODE 30
. WRITE BUF_29 TO NODE 70
ENDIF
```

END

Figure 6-78

### 6.6.5 Node 21

#### 6.6.5.1 General Description

Node 21 simulates those peripherals used by the other nodes and is connected to the main system bus. Specifically, the MAD, ESM, and radar peripherals are simulated. (Radar and ESM data are not presently used by any node, but short-term plans include the development of a radar/ESM contact correlation algorithm for use by an AN/AYK-14; therefore, these peripherals have been included in the simulation, with Node 30 chosen as the destination of the simulated data.

Node 21 will also contain the JTIDS simulation currently being developed for BASIC by 503 division.

Finally, Node 21 will simulate the reconfiguration bulk store and will transmit reconfiguration programs over the bus, as required.

#### 6.6.5.2 I/O Description

The input and output messages making up the Node 21 interface are shown in Tables 6-17 and 6-18, respectively. An asynchronous command output has no present use but has been included for flexibility.

#### 6.6.5.3 Software Functional Description

6.6.5.3.1 Control Modules. The control module requirements for Node 21 are identical to those of Node 70. Figures 6-79 through 6-83 describe these software executive functions.

6.6.5.3.2 Operational Modules. Node 21 simply outputs preset data according to Table 6-18 and accepts whatever data is sent to it. This node executes a limited number of commands from Node 82 to Node 90. Figures 6-84 through 6-88 describe these procedures.

6.6.5.3.3 Test and Reconfiguration Modules. Node 21 outputs a self-test message every minor cycle and also responds to a read block command from Node 90. This command will cause a reconfiguration program to be read from bulk storage and transmitted over the bus. This process is described by Figures 6-89 and 6-90.

TABLE 6-17. NODE 21 INPUT MESSAGES

	SRC NODE	RCV SUB ADR	WORDS PER SECOND	PERIOD IN MSEC	CYC CODE	WORD COUNT
***						
*	70	1	210	100	1	21
*	70	23	960	1000	0	32
*	82	30	6	0		3
*	90	29	160	50	0	8
*	90	30		0		7

TABLE 6-18. NODE 21 OUTPUT MESSAGES

DEST. NODE	XMT SUB ADR	DEST. SUBSYSTEM	TYPE	FORMAT	CYCLE CODE	WORD COUNT	RCV SUB ADR	CONTENTS
70	2	NAV	Static	Bit String	0	32	2	Simulated JTIDS Data
	3		Static	Bit String	0	32	3	
	4		Static	Bit String	0	32	4	
	5		Static	Bit String	0	32	5	
	6		Static	Bit String	1	32	6	
	7		Static	Bit String	2	16	7	
	8		Static	Bit String	2	30	1	
117	8	COMM	Static	Bit String	2	30	1	COMM XMTR
30	14	Radar (Simulated)	Static	Bit String	0	3	22	Simulated Radar Input (Node 105). (Node 30 will accept but not use)
	17		Static	Bit String	4	8	24	
82	17	Display	Static	Bit String	4	8	24	Simulated SRS Data (Node 108)
30	20	ESM (Simulated)	Static	Bit String	2	24	10	Simulated ESM Data (Node 111) (Node 30 will accept but not use)
	24		Static	Bit String	0	3	3	
30	24	MAD	Static	Bit String	0	3	3	Simulated MAD Data
82	30	Display	Command	CMD ID Data Word Data Word	-	3	30	Alert

NADC-79161-40



NADC-79161-40

Figures 6-79 through 6-83. Node 21 Executive Functions

NADC-79161-40  
MODULE NAME: INITIALIZE PERIPHERAL SIMULATION

DATA REQUIRED

NONE

DATA PRODUCED

NONE

INITIALIZE PERIPHERAL SIMULATION

THIS PROCESS IS ACTUATED BY A POWER ON CONDITION (IF POSSIBLE). IT LOADS PROGRAMS AND PRESET DATA FROM THE FLOPPY DISK AND SETS THE NECESSARY INITIAL VALUES. IT ALSO INITIALIZES THE INTERRUPT STRUCTURE AND SCHEDULES THE OUTPUT PROCESSING AND SELF TEST MODULES. FINALLY, IT CALLS THE "DISPATCH" MODULE TO INITIATE PROCESSING.

END

Figure 6-79

MODULE NAME: INTERRUPT HANDLER

DATA REQUIRED

TYPE OF EVENT (EXTERNAL OR BUS INPUT)

DATA PRODUCED

NONE

INTERRUPT HANDLER

THIS FUNCTION SAVES THE STATE OF THE MACHINE AND TRANSFERS CONTROL TO THE PROPER INTERRUPT RESPONSE MODULE

LOCK OUT ALL FURTHER INTERRUPTS  
SAVE PROGRAM CTR  
SAVE STACK POINTERS  
SAVE REGISTER CONTENTS  
CASE INTERRUPT TYPE (EXTERNAL OR BUS INPUT)  
  . "EXTERNAL INTERRUPT"  
  . "READ"  
ENDCASE  
RESTORE MACHINE STATE  
ENABLE INTERRUPTS  
TRANSFER TO PROGRAM CTR  
END

Figure 6-80

MODULE NAME: DISPATCH

DATA REQUIRED

INIT FLAG (INITIALLY PRESET TO 0)  
 NEW\_MINOR\_CYCLE\_FLAG  
 PERIODIC PROGRAM PARAMETERS  
 -STACK\_POINTER  
 -DEPTH\_OF\_STACK

DATA PRODUCED

NONE

DISPATCH

THIS PROCESS CALLS ALL THE PERIODIC PROCESSES  
 EXECUTED BY THE Z-2. IT IS THE LOWEST PRIORITY  
 PROCESS AND IS PRE-EMPTED BY ALL INTERRUPTS.  
 IT REPEATS FOR EACH MINOR CYCLE. PERIODIC PROCESS  
 ARE PRIORITIZED BY THEIR POSITION IN THE STACK.

DOWHILE POWER IS ON  
 . IF NEW\_MINOR\_CYCLE\_FLAG EQ 1 AND INIT\_FLAG EQ 1  
 . . THEN  
 . . . SET NEW\_MINOR\_CYCLE\_FLAG TO 0  
 . . . DOWHILE STACK NOT EMPTY  
 . . . . INITIALIZE MACHINE STATE FOR EACH PERIODIC PROCESS.  
 . . . . EACH STACK ENTRY POINTS TO A BUFFER AREA CONTAINING  
 . . . . . INITIALIZATION PARAMETERS.  
 . . . . . CALL PROCESS REPRESENTED BY STACK ENTRY.  
 . . ENDDO  
 . ENDIF  
 ENDDO  
 END

Figure 6-81

NADC-79161-40

MODULE NAME: READ

DATA REQUIRED

INPUT MESSAGE (ASSUMED TO BE LOCATED IN I.M. BUFFER)  
TABLE OF MEMORY BUFFER LOCATIONS FOR EACH RCV ADDRESS.  
TABLE OF BUFFER LENGTHS FOR EACH RCV ADDRESS.  
INPUT MESSAGE RCV ADDRESS

DATA PRODUCED

INPUT MESSAGE IN RCV ADDRESS BUFFER

READ

THIS MODULE INPUTS MESSAGES FROM THE INTERFACE MODULE

IF RCV ADDRESS EQ 30  
  . THEN  
  . INPUT MESSAGE TO RCV ADR 30  
  . ACKNOWLEDGE INPUT  
  . CALL "COMMAND PROCESSING" MODULE  
  . ELSE  
  . TRANSFER MESSAGE TO APPROPRIATE INPUT BUFFER  
  . ACKNOWLEDGE INPUT  
ENDIF  
END

Figure 6-82

MODULE NAME: WRITE

DATA REQUIRED

XMT MESSAGE ADDRESS  
TABLE OF BUFFER LOCATIONS FOR EACH XMT ADDRESS  
TABLE OF BUFFER LENGTHS FOR EACH XMT ADDRESS

DATA PRODUCED

OUTPUT MESSAGE TO INTERFACE MODULE

WRITE

THIS FUNCTION OUTPUTS THE SPECIFIED MESSAGE  
TO THE INTERFACE MODULE. IT WAITS FOR AN ACKNOWLEDGE  
FOR A SPECIFIED PERIOD OF TIME. NONE OCCURRING, AN  
ERROR CONDITION IS NOTED IN THE STATUS WORD.

END

Figure 6-83



NADC-79161-40

Figures 6-84 through 6-88. Node 21 Operational Functions

MODULE NAME: PERIPHERAL OUTPUT SIMULATION

DATA REQUIRED

PRESET DATA FOR OUTPUT BUFFERS  
-OUTPUT BUFFERS DO NOT CHANGE IF DATA IS STATIC.  
-IF DATA IS VARIED IN A PREDEFINED MANNER,  
-A SEQUENCE OF PRESET BLOCKS IS DEFINED  
-WHICH CAN BE ACCESSED USING A SEQUENCE NUMBER  
-CONTAINED IN BUF\_SEQ, WHICH IS A VECTOR CONTAINING  
-THE CURRENT SEQUENCE NUMBER FOR EACH BUFFER.  
-THE LENGTH OF EACH SEQUENCE IS CONTAINED IN  
-THE VECTOR MAX\_BUF\_SEQ

DATA PRODUCED

UPDATES OF OUTPUT BUFFERS FOR TRANSMISSION  
BUF\_2 (XMT SUB ADDRESS 2- 32 WORDS)  
BUF\_3 -32 WDS  
BUF\_4-32 WDS  
BUF\_5-32 WDS  
BUF\_6-32 WDS  
BUF\_7-16 WDS  
BUF\_8-16 WDS  
BUF\_14-3 WDS  
BUF\_17-8 WDS  
BUF\_20-24 WDS  
BUF\_24-3 WDS

Figure 6-84a

## PERIPHERAL OUTPUT SIMULATION

THIS MODULE EXECUTES EVERY MINOR CYCLE

```

INCREMENT CYCLE_CODE_1_FLAG
INCREMENT CYCLE_CODE_2_FLAG
INCREMENT CYCLE_CODE_4_FLAG
IF PER_INIT_FLAG(2) EQ 1
. IF BUF_2 MUST BE VARIED ACCORDING TO A PRESET SEQUENCE
. . TRANSFER 32 WORDS TO BUF_2 FROM AREA INDICATED BY BUF_SEQ(2)
. . INCREMENT BUF_SEQ(2)
. . IF BUF_SEQ(2) GT MAX_BUF_SEQ(2)
. . . SET BUF_SEQ(2) TO 0
. . ENDIF
. . WRITE BUF_2 TO INTERFACE MODULE
. ENDIF
ENDIF
IF PER_INIT_FLAG(3) EQ 1
. IF BUF_3 MUST BE VARIED ACCORDING TO A PRESET SEQUENCE
. . TRANSFER 32 WORDS TO BUF_3 FROM AREA INDICATED BY BUF_SEQ(3)
. . INCREMENT BUF_SEQ(3)
. . IF BUF_SEQ(3) GT MAX_BUF_SEQ(3)
. . . SET BUF_SEQ(3) TO 0
. . ENDIF
. . WRITE BUF_3 TO INTERFACE MODULE
. ENDIF
ENDIF
IF PER_INIT_FLAG(4) EQ 1
. IF BUF_4 MUST BE VARIED ACCORDING TO A PRESET SEQUENCE
. . TRANSFER 32 WORDS TO BUF_4 FROM AREA INDICATED BY BUF_SEQ(4)
. . INCREMENT BUF_SEQ(4)
. . IF BUF_SEQ(4) GT MAX_BUF_SEQ(4)
. . . SET BUF_SEQ(4) TO 0
. . ENDIF
. . WRITE BUF_4 TO INTERFACE MODULE
. ENDIF
ENDIF
IF PER_INIT_FLAG(5) EQ 1
. IF BUF_5 MUST BE VARIED ACCORDING TO A PRESET SEQUENCE
. . TRANSFER 32 WORDS TO BUF_5 FROM AREA INDICATED BY BUF_SEQ(5)
. . INCREMENT BUF_SEQ(5)
. . IF BUF_SEQ(5) GT MAX_BUF_SEQ(5)
. . . SET BUF_SEQ(5) TO 0
. . ENDIF
. . WRITE BUF_5 TO INTERFACE MODULE
. ENDIF
ENDIF
IF PER_INIT_FLAG(6) EQ 1
. IF CYCLE_CODE_1_FLAG EQ 2
. . SET CYCLE_CODE_1_FLAG TO 0
. . IF BUF_6 MUST BE VARIED ACCORDING TO A PRESET SEQUENCE
. . . TRANSFER 32 WORDS TO BUF_6 FROM AREA INDICATED BY BUF_SEQ(6)
. . . INCREMENT BUF_SEQ(6)
. . . IF BUF_SEQ(6) GT MAX_BUF_SEQ(6)
. . . . SET BUF_SEQ(6) TO 0
. . . ENDIF
. . . WRITE BUF_6 TO INTERFACE MODULE
. . . ENDIF
. ENDIF
ENDIF
IF PER_INIT_FLAG(7) EQ 1

```

Figure 6-84b

```

. IF CYCLE_CODE_2_FLAG EQ 4
. . SET CYCLE_CODE_2_FLAG TO 0
. . IF BUF_7 MUST BE VARIED ACCORDING TO A PRESET SEQUENCE
. . . TRANSFER 16 WORDS TO BUF_7 FROM AREA INDICATED BY BUF_SEQ(7)
. . . INCREMENT BUF_SEQ(7)
. . . IF BUF_SEQ(7) GT MAX_BUF_SEQ(7)
. . . . SET BUF_SEQ(7) TO 0
. . . ENDIF
. . WRITE BUF_7 TO INTERFACE MODULE
. . ENDF
. ENDF
ENDIF
IF PER_INIT_FLAG(8) EQ 1
. IF CYCLE_CODE_2_FLAG EQ 4
. . SET CYCLE_CODE_2_FLAG TO 0
. . IF BUF_8 MUST BE VARIED ACCORDING TO A PRESET SEQUENCE
. . . TRANSFER 30 WORDS TO BUF_8 FROM AREA INDICATED BY BUF_SEQ(8)
. . . INCREMENT BUF_SEQ(8)
. . . IF BUF_SEQ(8) GT MAX_BUF_SEQ(8)
. . . . SET BUF_SEQ(8) TO 0
. . . ENDIF
. . WRITE BUF_8 TO INTERFACE MODULE
. . ENDF
. ENDF
ENDIF
IF PER_INIT_FLAG(14) EQ 1
. IF BUF_14 MUST BE VARIED ACCORDING TO A PRESET SEQUENCE
. . TRANSFER 3 WORDS TO BUF_14 FROM AREA INDICATED BY BUF_SEQ(14)
. . INCREMENT BUF_SEQ(14)
. . IF BUF_SEQ(14) GT MAX_BUF_SEQ(14)
. . . SET BUF_SEQ(14) TO 0
. . . ENDIF
. . WRITE BUF_14 TO INTERFACE MODULE
. . ENDF
ENDIF
IF PER_INIT_FLAG(17) EQ 1
. IF CYCLE_CODE_4_FLAG EQ 16
. . SET CYCLE_CODE_4_FLAG TO 0
. . IF BUF_17 MUST BE VARIED ACCORDING TO A PRESET SEQUENCE
. . . TRANSFER 9 WORDS TO BUF_17 FROM AREA INDICATED BY BUF_SEQ(17)
. . . INCREMENT BUF_SEQ(17)
. . . IF BUF_SEQ(17) GT MAX_BUF_SEQ(17)
. . . . SET BUF_SEQ(17) TO 0
. . . ENDIF
. . WRITE BUF_17 TO INTERFACE MODULE
. . ENDF
. ENDF
ENDIF
IF PER_INIT_FLAG(20) EQ 1
. IF CYCLE_CODE_2_FLAG EQ 4
. . SET CYCLE_CODE_2_FLAG TO 0
. . IF BUF_20 MUST BE VARIED ACCORDING TO A PRESET SEQUENCE
. . . TRANSFER 24 WORDS TO BUF_20 FROM AREA INDICATED BY BUF_SEQ(20)
. . . INCREMENT BUF_SEQ(20)
. . . IF BUF_SEQ(20) GT MAX_BUF_SEQ(20)
. . . . SET BUF_SEQ(20) TO 0
. . . ENDIF
. . WRITE BUF_20 TO INTERFACE MODULE
. . ENDF
. ENDF
ENDIF
IF PER_INIT_FLAG(24) EQ 1
. IF BUF_24 MUST BE VARIED ACCORDING TO A PRESET SEQUENCE
. . TRANSFER 3 WORDS TO BUF_24 FROM AREA INDICATED BY BUF_SEQ(24)

```

Figure 6-84c



NADC-79161-40

```
. . INCREMENT BUF_SEQ(24)
. . IF BUF_SEQ(24) GT MAX_BUF_SEQ(24)
. . . SET BUF_SEQ(24) TO 0
. . ENDIF
. . WRITE BUF_24 TO INTERFACE MODULE
. . ENDIF
ENDIF
END
```

Figure 6-84d

MODULE NAME: COMMAND PROCESSING

DATA REQUIRED

INPUT COMMAND ID AND DATA WORDS FROM RCV ADDRESS 30 INPUT BUFFER

DATA PRODUCED

NONE

COMMAND PROCESSING

THIS MODULE INTERPRETS THE COMMAND ID AND CALLS THE APPROPRIATE  
SUBROUTINE TO EXECUTE THE COMMAND

```
CASE COMMAND ID
. $COMM INIT$
. $NEW MINOR CYCLES$
. $READ BLOCK$
. $COMM TERM$
ENDCASE
```

END

Figure 6-85

NADC-79161-40

```

COMM INIT
SET INIT_FLAG TO 1
FORM 32 BIT INPUT DATA WORD FROM DATA WORDS 1 AND 2 OF INPUT BUFFER 30
SET I TO 1
DOWHILE I LE 32
. SET PER_INIT_FLAG(I) TO BIT(I) OF INPUT DATA WORD
. INCREMENT I
ENDDO
END

```

Figure 6-86

```

NEW MINOR CYCLE
SET NEW_MINOR_CYCLE_FLAG TO 1
END

```

Figure 6-87

```

COMM TERM
SET INIT_FLAG TO 0
END

```

Figure 6-88

NADC-79161-40

Figures 6-89 and 6-90. Node 21 Test and Reconfiguration Function

READ BLOCK

```

SET ADR TO FIRST WORD ADDRESS OF BLOCK AS SPECIFIED BY INPUT DATA WORD
SET BLOCK_LENGTH TO VALUE SPECIFIED BY INPUT WORD
SET DESTINATION NODE ACCORDING TO INPUT DATA
SET I TO 1
SET J TO 1
SET DONE TO 0
DOWHILE I LE BLOCK_LENGTH AND DONE EQ 0
. SET J TO 1
. DOWHILE J LE 32 AND DONE EQ 0
. . TRANSFER MEMORY WORD(ADR) TO XMT SUB ADDRESS 30 , WORD J
. . INCREMENT I
. . INCREMENT J
. . INCREMENT ADR
. . IF I GT BLOCK_LENGTH
. . . SET DONE TO 1
. . ENDF
. ENDDO
. WRITE XMT SUB ADDRESS 30 BUFFER TO DESTINATION NODE
ENDDO
END

```

Figure 6-89



NADC-79161-40

MODULE NAME PERIPHERAL SIMULATION SELF TEST

DATA REQUIRED

RESULTS OF SELF TEST PROGRAM  
2 STATUS WORDS  
OLD SEQUENCE WORD (SEQUENCE WORD LAST TRANSMITTED)  
LAST RETURN WORD INPUT FROM NODE 90

DATA PRODUCED

8 WORD SELF TEST MESSAGE (BUF\_29)  
-NODE ID, MAJOR CYCLE NO, MINOR CYCLE NO  
-STATUS WORD 1  
-STATUS WORD 2  
-CONFIGURATION IDENT  
-RESULTS OF SELF TEST PROGRAM  
-OLD SEQUENCE WORD  
-RETURN WORD  
-NEW SEQUENCE WORD

PERIPHERAL SIMULATION SELF TEST

THIS MODULE EXECUTES EVERY MINOR CYCLE

SET WORD 5 OF BUF\_29 TO RESULT OF SELF TEST PROGRAM  
IF WORD 5 INDICATES A FAILURE  
  . THEN  
  . IF FAILURE RECOVERABLE  
  . . THEN  
  . . CALL APPROPRIATE RECOVERY ROUTINE  
  . . SET STATUS WORDS AND CONFIGURATION ID TO INDICATE NEW  
  . . CONFIGURATION  
  . . ELSE  
  . . ATTEMPT TO ACTIVATE SHUTDOWN MECHANISM  
  . ENDOF  
  . ELSE  
  . IF NEW MAJOR CYCLE  
  . . INCREMENT MAJOR CYCLE NO.  
  . . SET MINOR CYCLE NO. TO 0  
  . ENDOF  
  . SET WORD 1 OF BUF\_29 TO NODE ID, MAJOR CYCLE NO, MINOR CYCLE NO  
  . INCREMENT MINOR CYCLE NO.  
  . COPY STATUS WORD 1 INTO WORD 2 OF BUF\_29  
  . COPY STATUS WORD 2 INTO WORD 3 OF BUF\_29  
  . SET WORD 4 OF BUF\_29 TO CONFIGURATION ID  
  . SET WORD 5 OF BUF\_29 TO LAST WORD 8 TRANSMITTED FROM BUF\_29  
  . SET WORD 7 OF BUF\_29 TO LAST WORD 8 RECEIVED FROM NODE 90  
  . SET WORD 8 OF BUF\_29 TO A NEW SEQUENCE WORD  
  . WRITE BUF\_29 TO I.M.  
ENDIF  
END

Figure 6-90

REFERENCES

1. Sperry Univac, "Avionic Information Processing System Design Methodology, Final Report," Contract N62269-76-C-0345, March 1977.
2. Sperry Univac, "Technical Progress Report for S-3A Information Processing Architecture Study," Contract N62269-77-C-0427, 6 January 1978.
3. Sperry Univac, "S-3A Information Processing Architecture Study, Requirement Decomposition," Contract N62269-77-C-0427, 5 May 1978.
4. Sperry Univac, "S-3A Information Processing Architecture Study, System Configuration," Contract N62269-77-C-0427, 5 May 1978.
5. Weissberger, A. J., "Analysis of Multiple-Microprocessor System Architectures," Computer Design, June 1977.
6. Moss, D., "Multiprocessing Adds Muscle to  $\mu$ Ps," Electronic Design, 24 May 1978.
7. Rozsa, K., "Multiprocessing Boosts Microcomputer Power Dramatically," Electronic Design, 15 March 1978.
8. Schoeffler, J. D., and Rose, C. W., "Distributed Computer Intelligence for Data Acquisition and Control," IEEE Transactions on Nuclear Science, Vol. NS-21, No. 1, February 1976.
9. Forsdick, H. C., et al., "Operating Systems for Computer Networks," Computer, January 1978.
10. Stockenberg, J., and Van Dam, A., "Vertical Migration for Performance Enhancement in Layered Hardware/Firmware/Software Systems," Computer, May 1978.
11. Eckhouse, R. H., et al., "Issues in Distributed Processing — An Overview of Two Workshops," Computer, January 1978.
12. Enslow, P. H., "What is a Distributed Data Processing," Computer, January 1978.
13. Jensen, E. D., "The Honeywell Experimental Distributed Processor — An Overview," Computer, January 1978.

14. Morgan, D. E. M, "A Survey of Methods for Improving Computer Network Reliability and Availability," Computer, November 1977.
15. Gonzalez, M. J., "Future Directions in Computer Architecture," Computer, March 1978.
16. Svendsen, E. C., and Ream, D. L., "What's Different About the Hardware in Tactical Military Systems," AFIPS Conf. Proc., Vol. 42, NCC 73, June 1973.
17. Phillips, W. C., "What's Different About Tactical Executive Systems," AFIPS Conf. Proc., Vol. 42, NCC 73, June 1973.
18. Liebowitz, B. H., and Carson, J. H., "Tutorial Distributed Processing," Fall CompCon 77, September 1977, IEEE EH0127-1.
19. Nollf, T. O., "Improvements in Real-Time Distributed Control," Digest of Papers, Fall CompCon 77, September 1977, pp. 409a-409g.
20. "Advanced Integrated Display System (AIDS) System Specification for Advanced Development Model," AIDS-78-111, April 1978.
21. "V/STOL 'A' Avionics Functional Description," NAVAIRDEVCON Center Design Team, 7 July 1978.
22. The Computer Program Performance Specification for Standard Executive for Use with AN/UYK-20 and AN/AYK-14 Computers, NAVELEX 0967-LP-598-2710, 28 April 1978.

NADC-79161-40

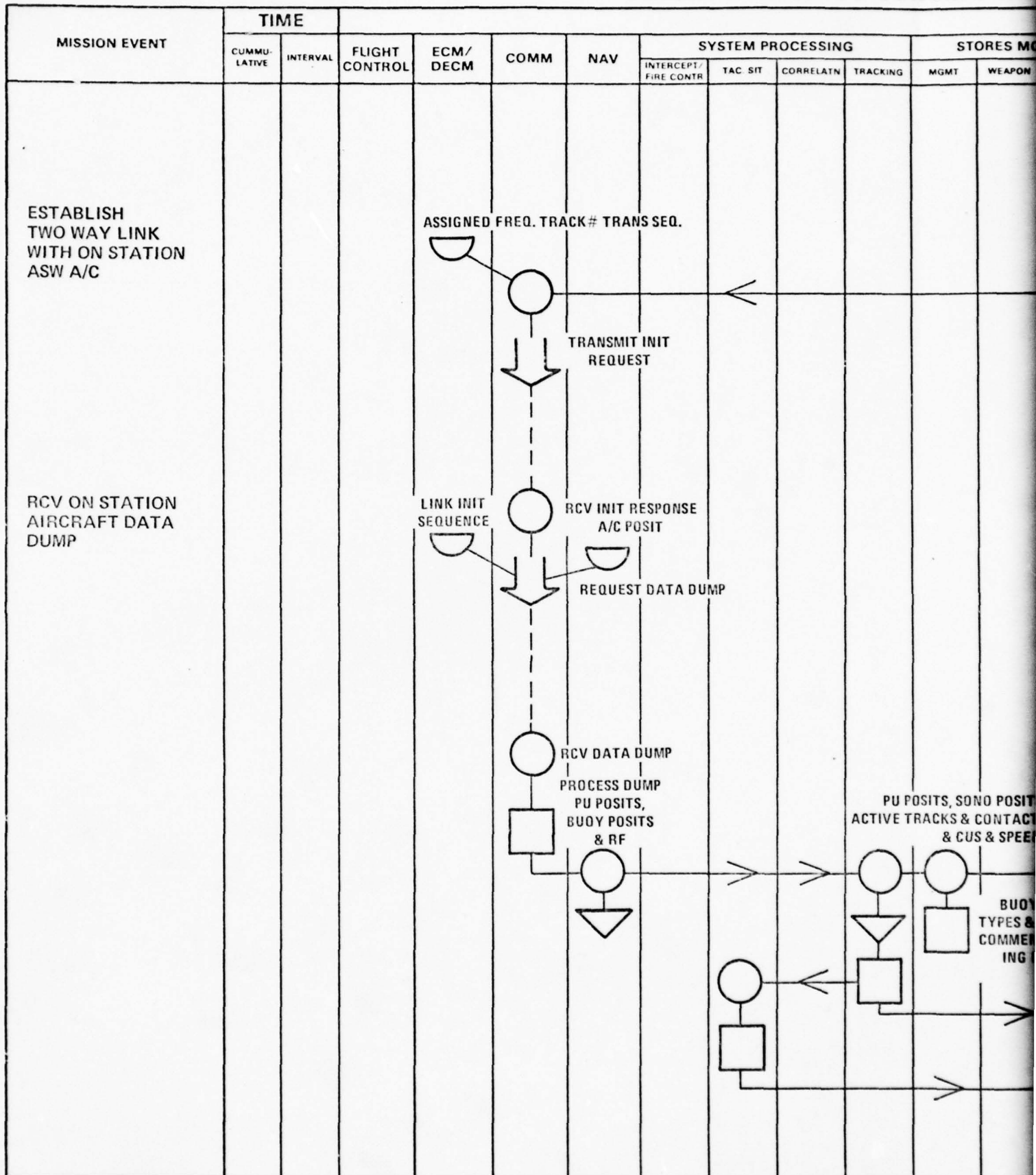
APPENDIX A IS NOT CONTAINED IN THIS VOLUME  
BUT IS AVAILABLE FROM NAVAIRDEVCON COMPU-  
TER SYSTEMS TECHNOLOGY DIVISION, CODE 502.



NADC-79161-40

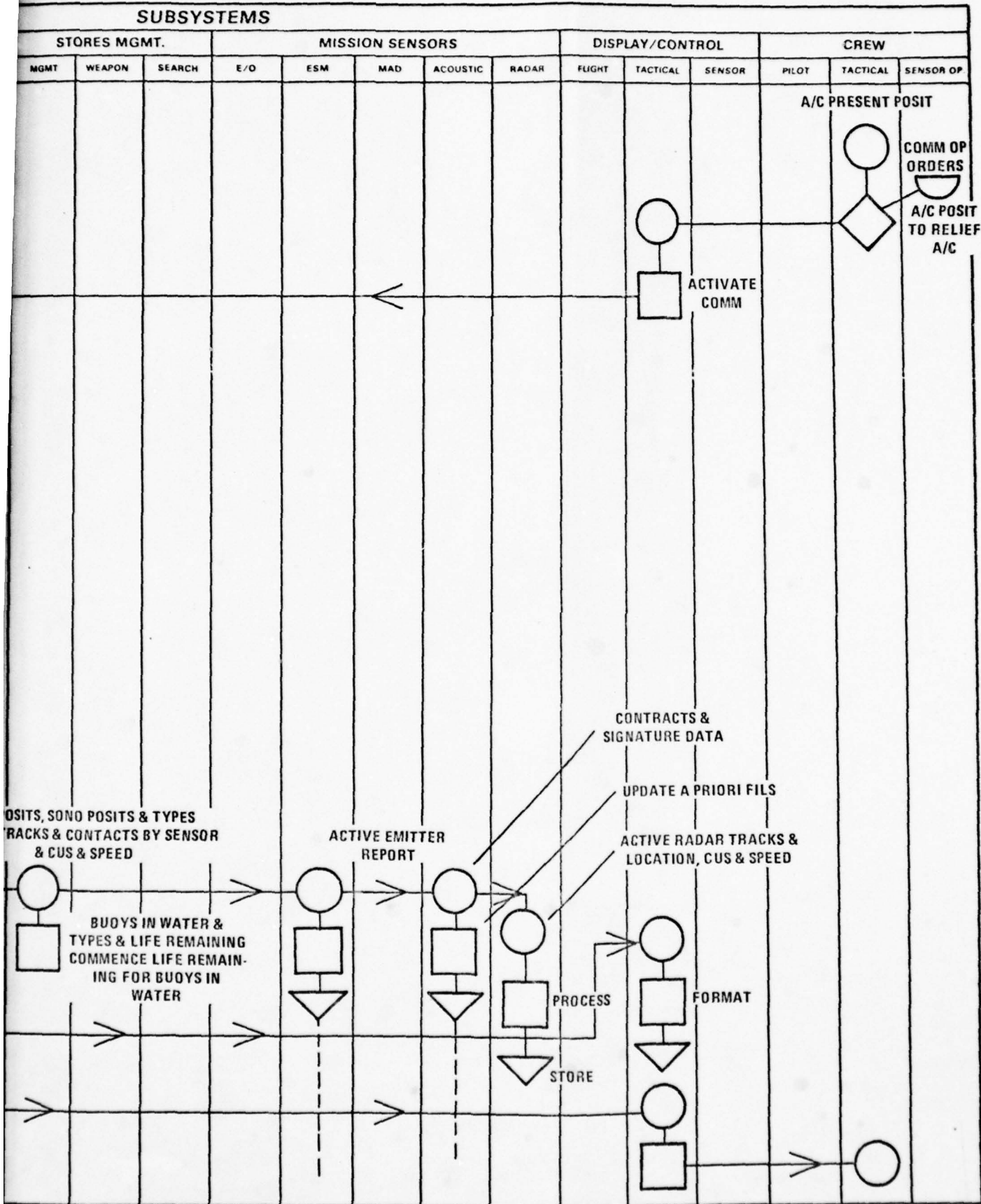
APPENDIX B  
OPERATIONAL SEQUENCE DIAGRAMS

# VSTOL F3 AVIONICS CONCEPT WITH



# CONCEPT WITH 3 MAN CREW

## SUBSYSTEMS

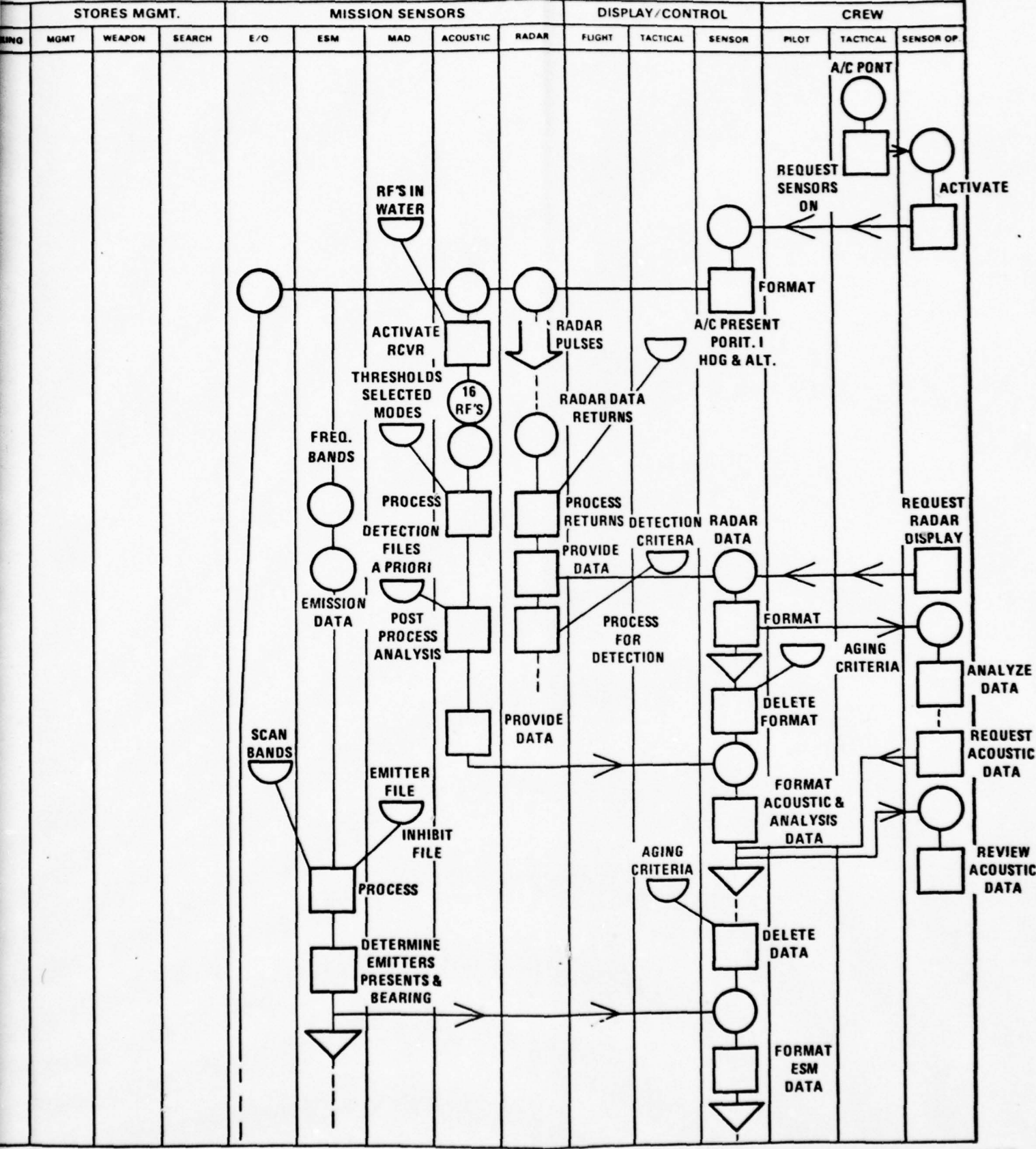


2



## CONCEPT WITH 3 MAN CREW

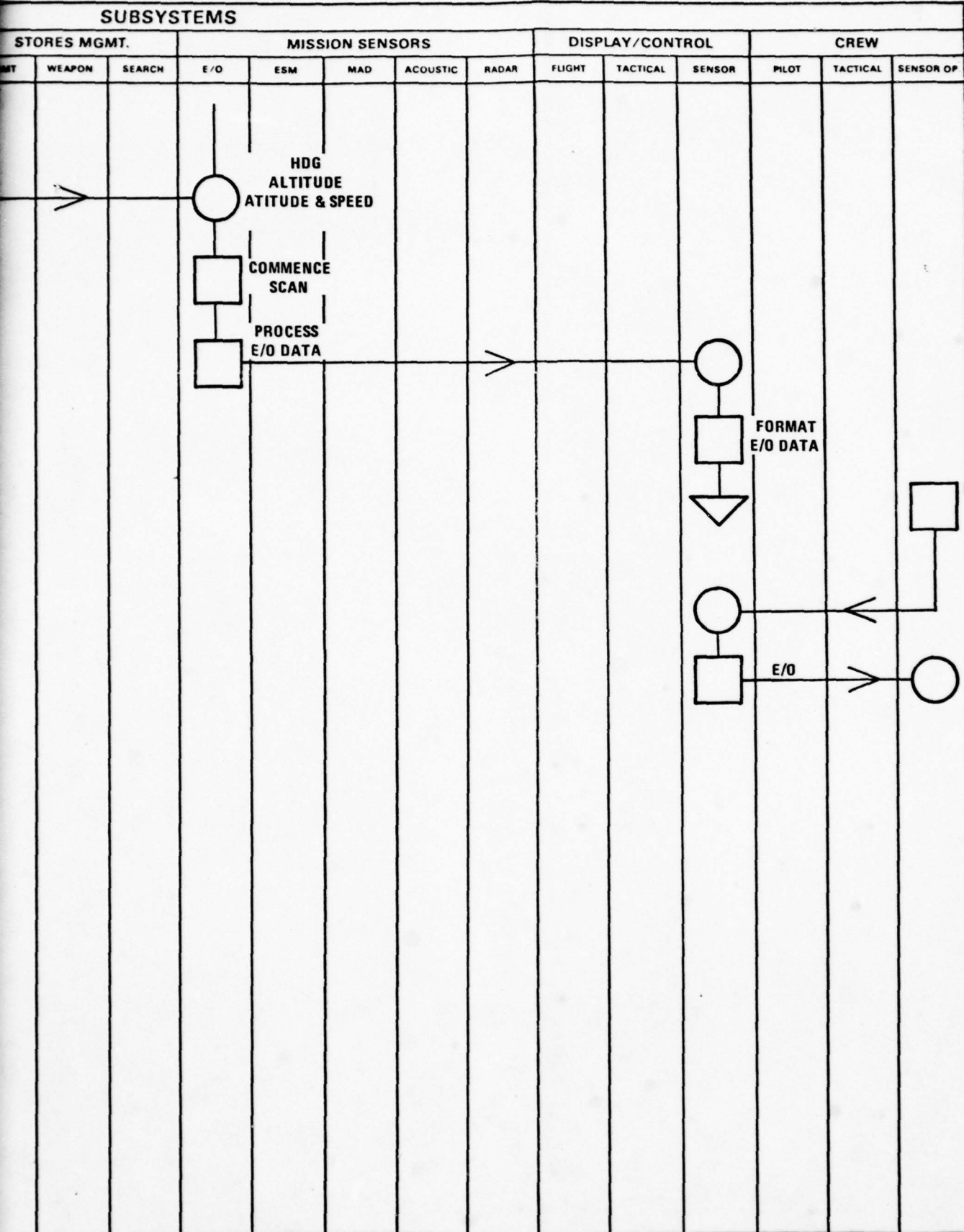
## SUBSYSTEMS



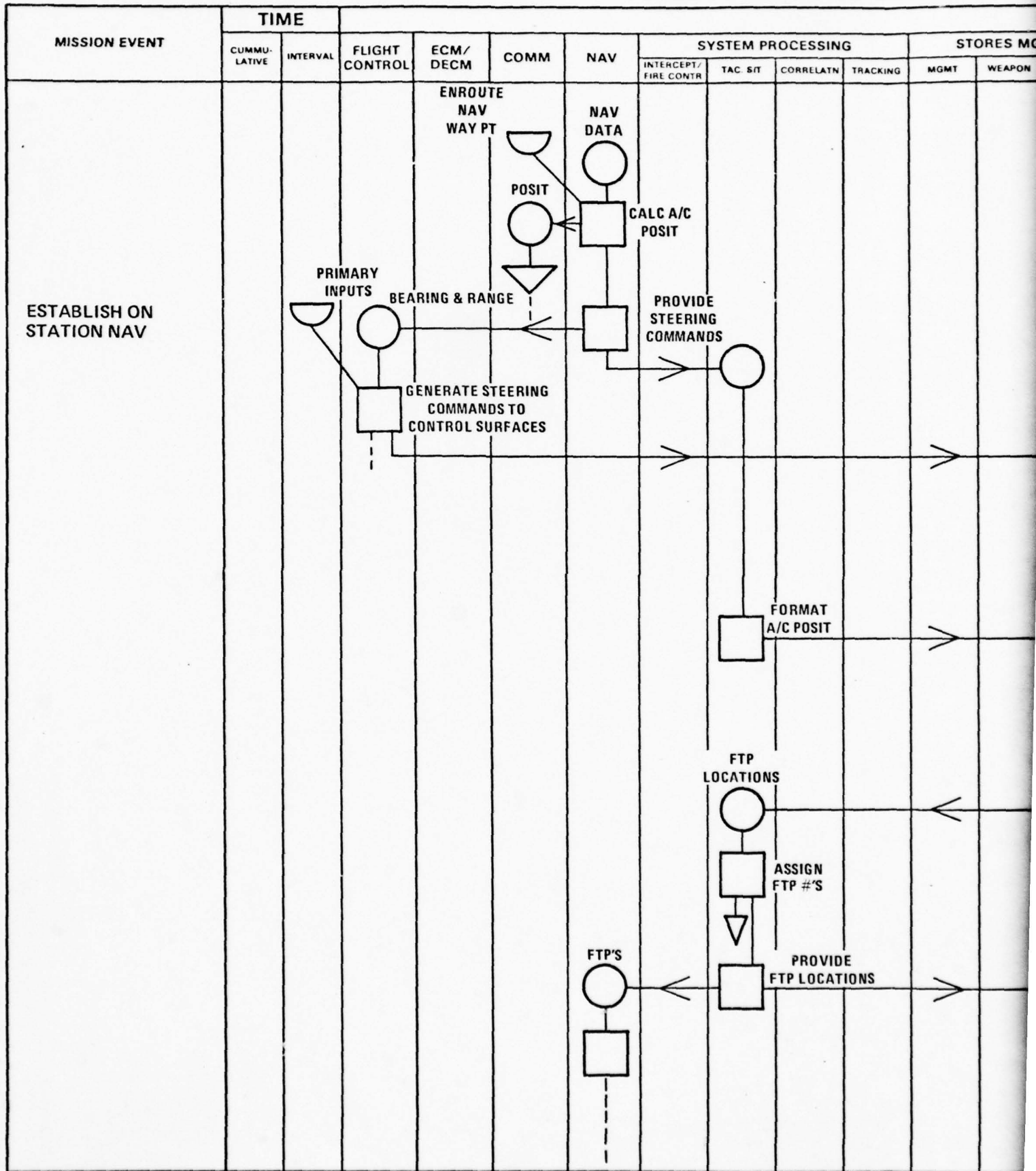
## VSTOL F3 AVIONICS CONCEPT WITH

[illegible]

## EPT WITH 3 MAN CREW



## VSTOL F3 AVIONICS CONCEPT WITH



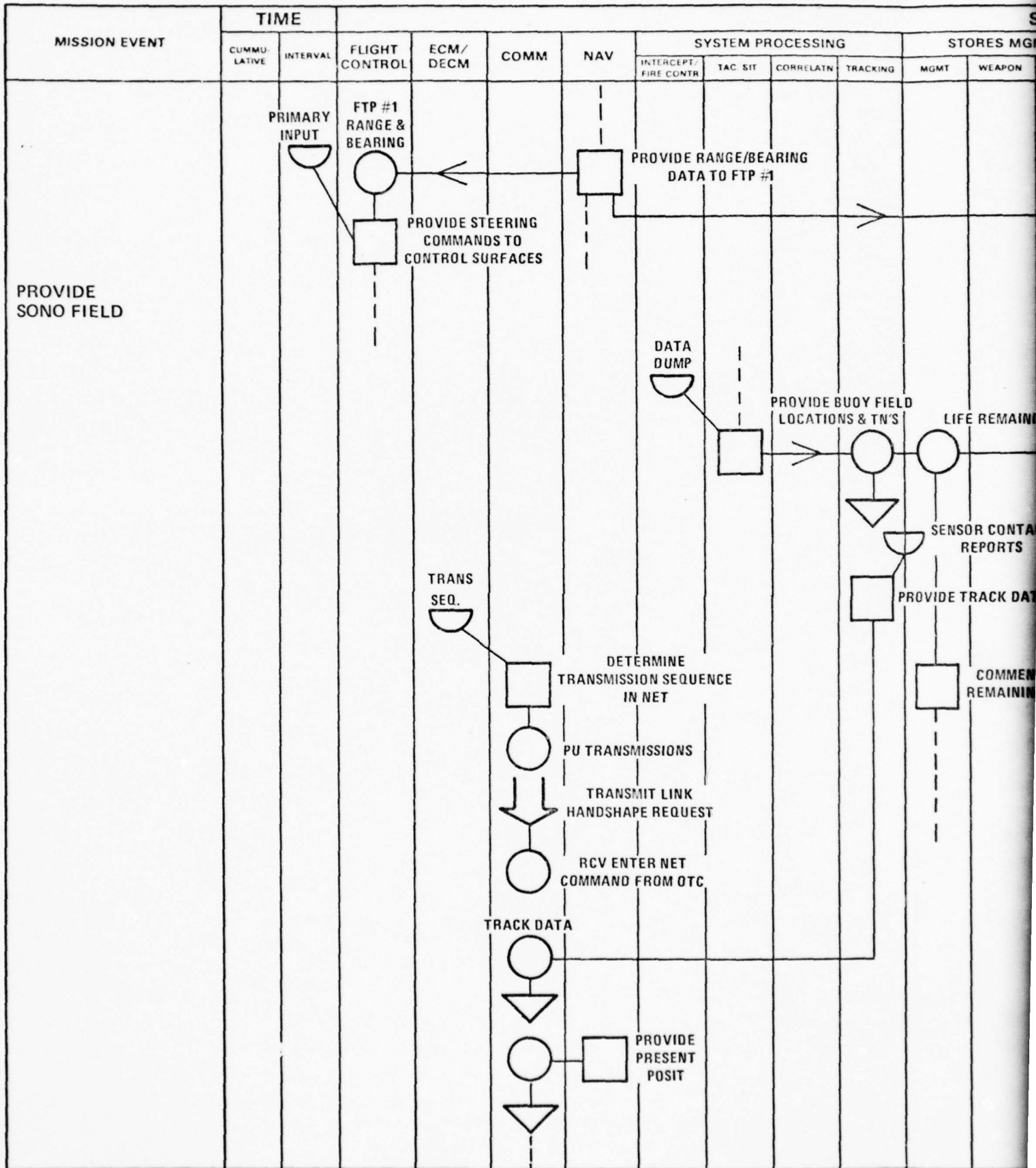


## CEPT WITH 3 MAN CREW

## SUBSYSTEMS

STORES MGMT.			MISSION SENSORS					DISPLAY/CONTROL			CREW		
MT	WEAPON	SEARCH	E/D	ESM	MAD	ACOUSTIC	RADAR	FLIGHT	TACTICAL	SENSOR	PILOT	TACTICAL	SENSOR OP
			>			>		A/C POSIT & FLIGHT DATA ○ □					
			>			>					○ □ PROVIDE PRIMARY INPUTS		
			>			>						A/C POSIT ○	INSERT SEARCH AREA FTP'S ◇
			<			<		ID FTP LOCATIONS □					
			>			>		FTP SYMBOLS ○ □					FTP'S ○

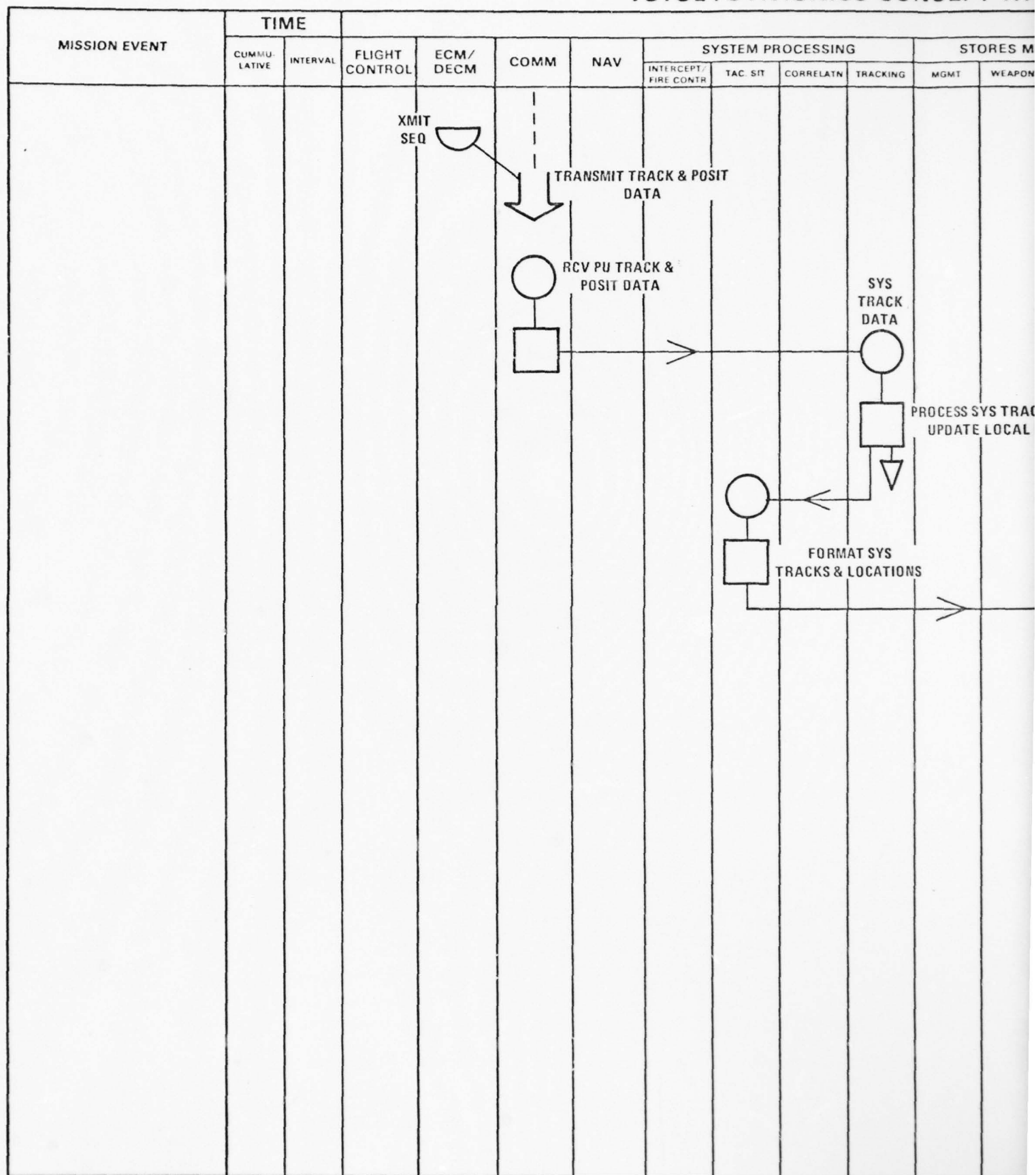
# VSTOL F3 AVIONICS CONCEPT WITH



## SUBSYSTEMS



## VSTOL F3 AVIONICS CONCEPT WITH





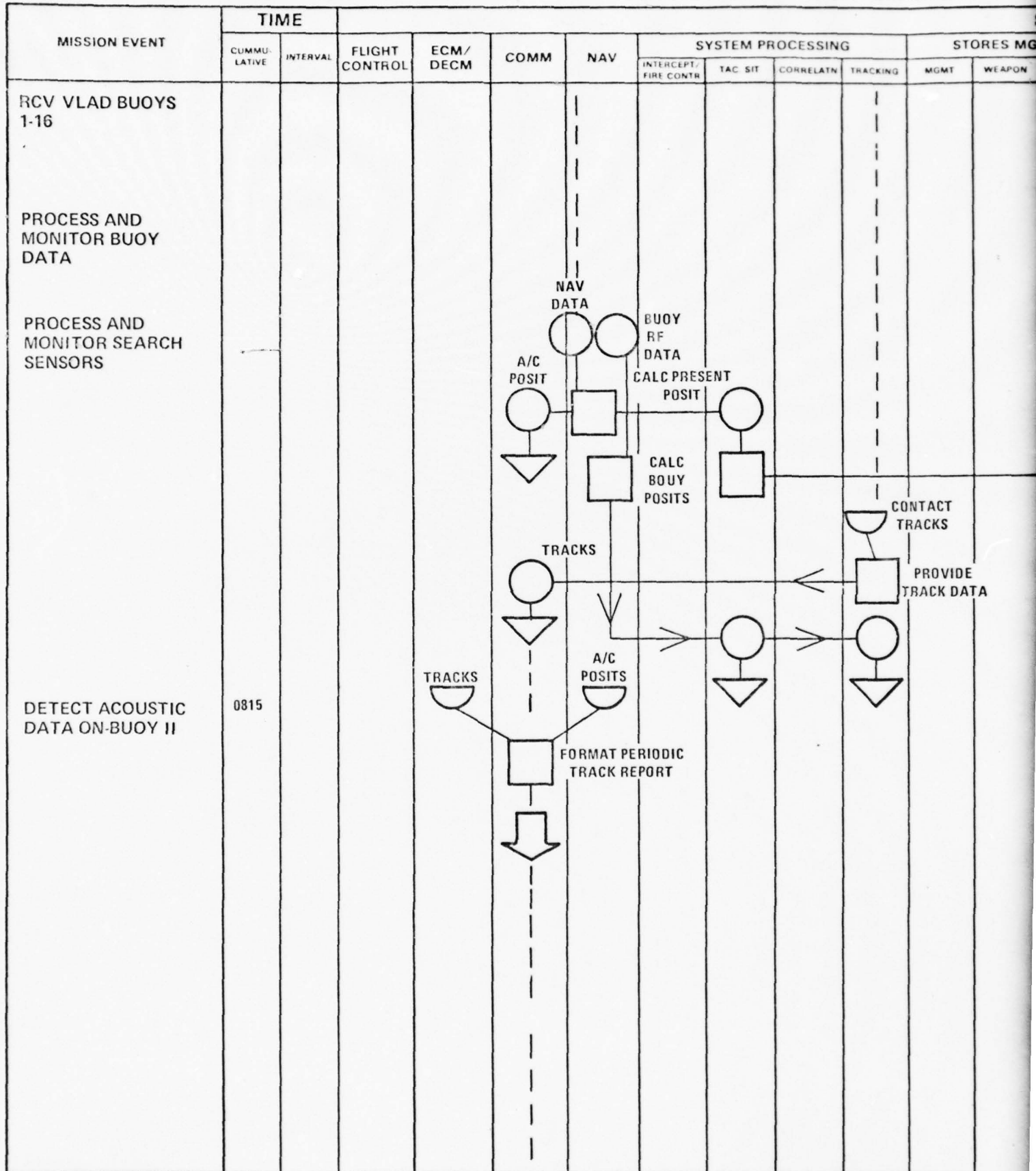
## CONCEPT WITH 3 MAN CREW

## SUBSYSTEMS

STORES MGMT.				MISSION SENSORS					DISPLAY/CONTROL			CREW		
MG	MGMT	WEAPON	SEARCH	E/O	ESM	MAD	ACOUSTIC	RADAR	FLIGHT	TACTICAL	SENSOR	PILOT	TACTICAL	SENSOR OP
<div> <div>PROCESS SYS TRACK FILE UPDATE LOCAL FILE</div> <div> <div>→</div> <div>→</div> <div> <div>○</div> <div>□</div> </div> <div>→</div> <div>○</div> </div> <div>REVIEW</div> </div>														

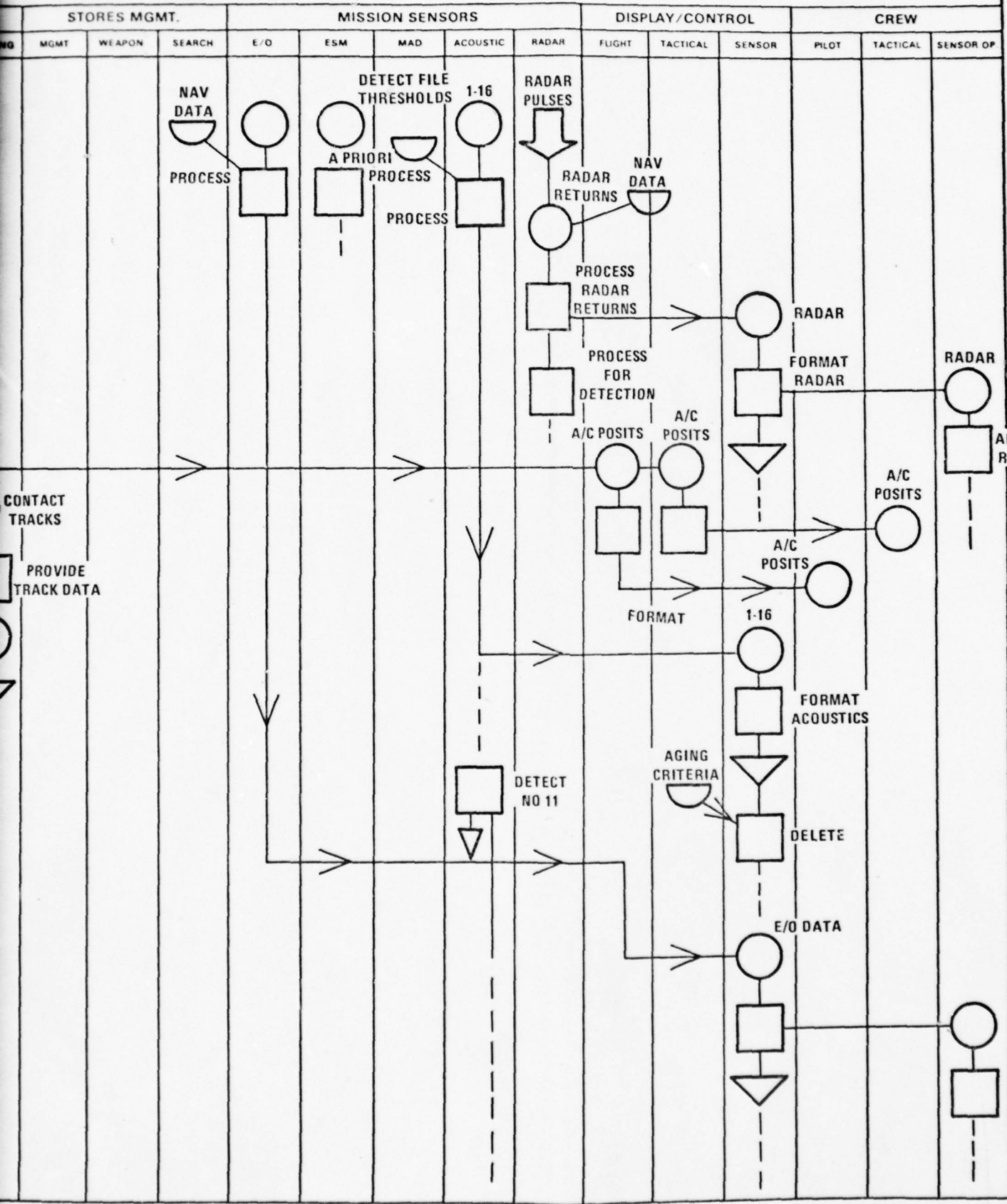
2

# VSTOL F3 AVIONICS CONCEPT WIT

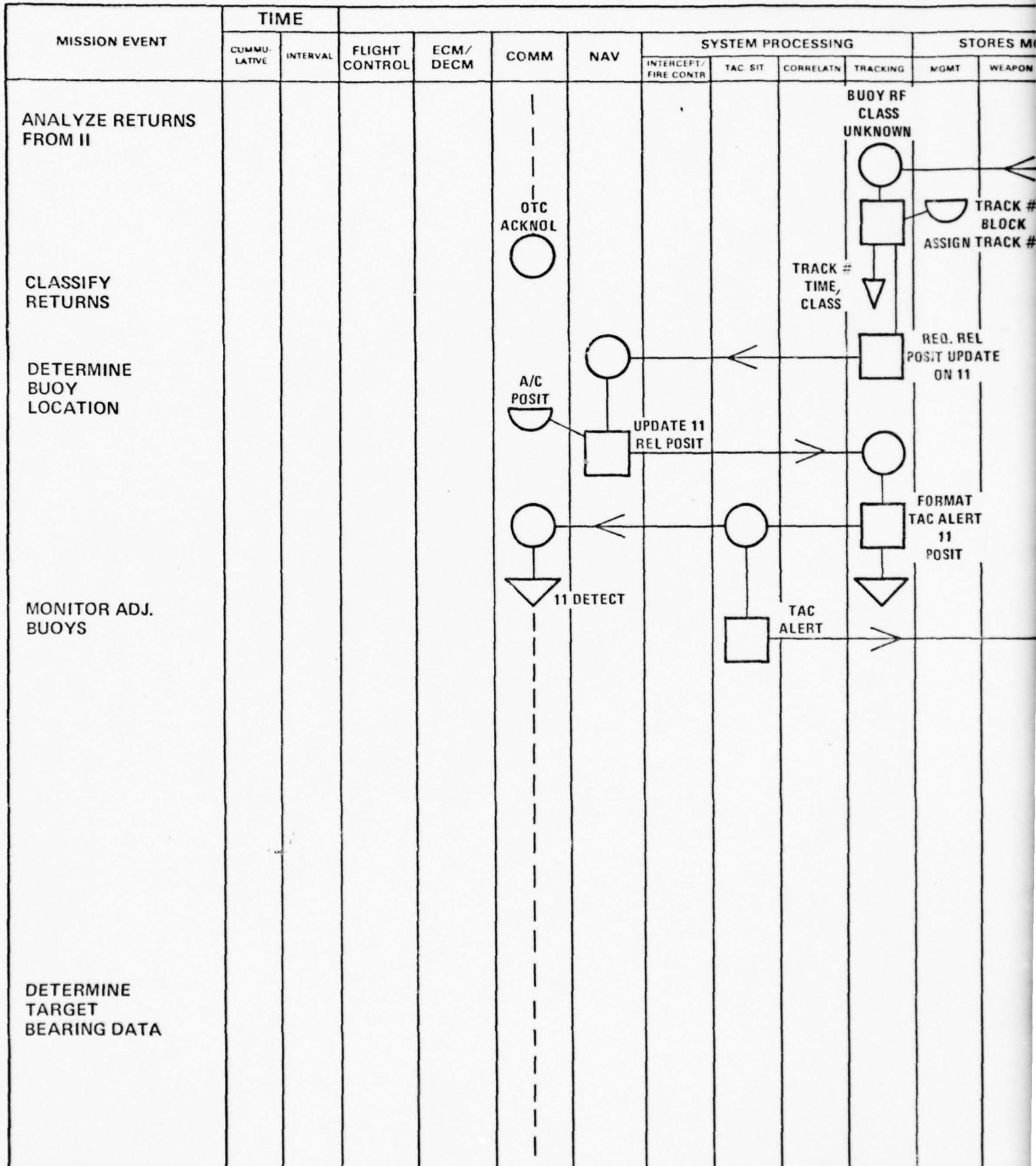


CONCEPT WITH 3 MAN CREW

SUBSYSTEMS



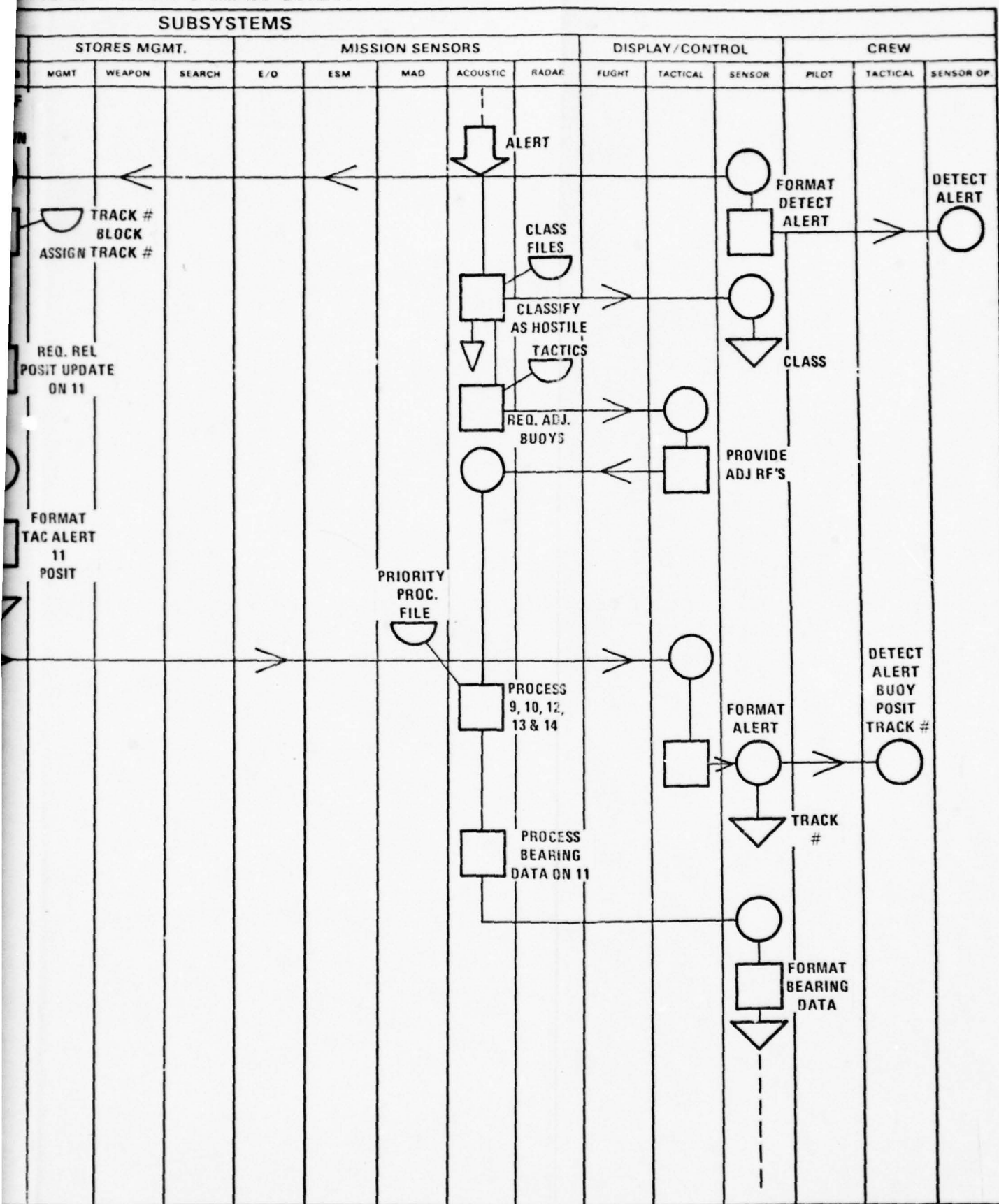
# VSTOL F3 AVIONICS CONCEPT WITH



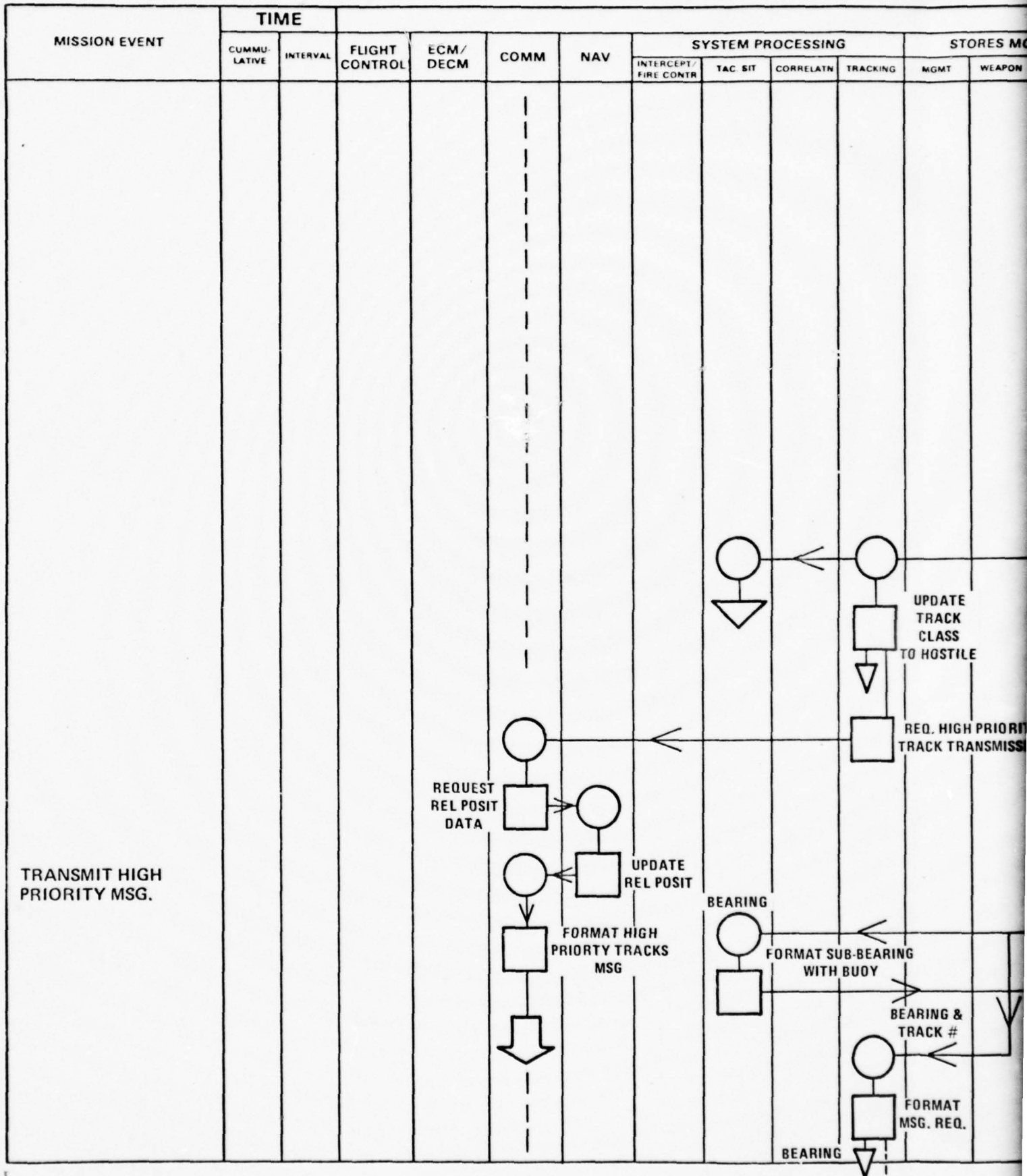


# CONCEPT WITH 3 MAN CREW

## SUBSYSTEMS

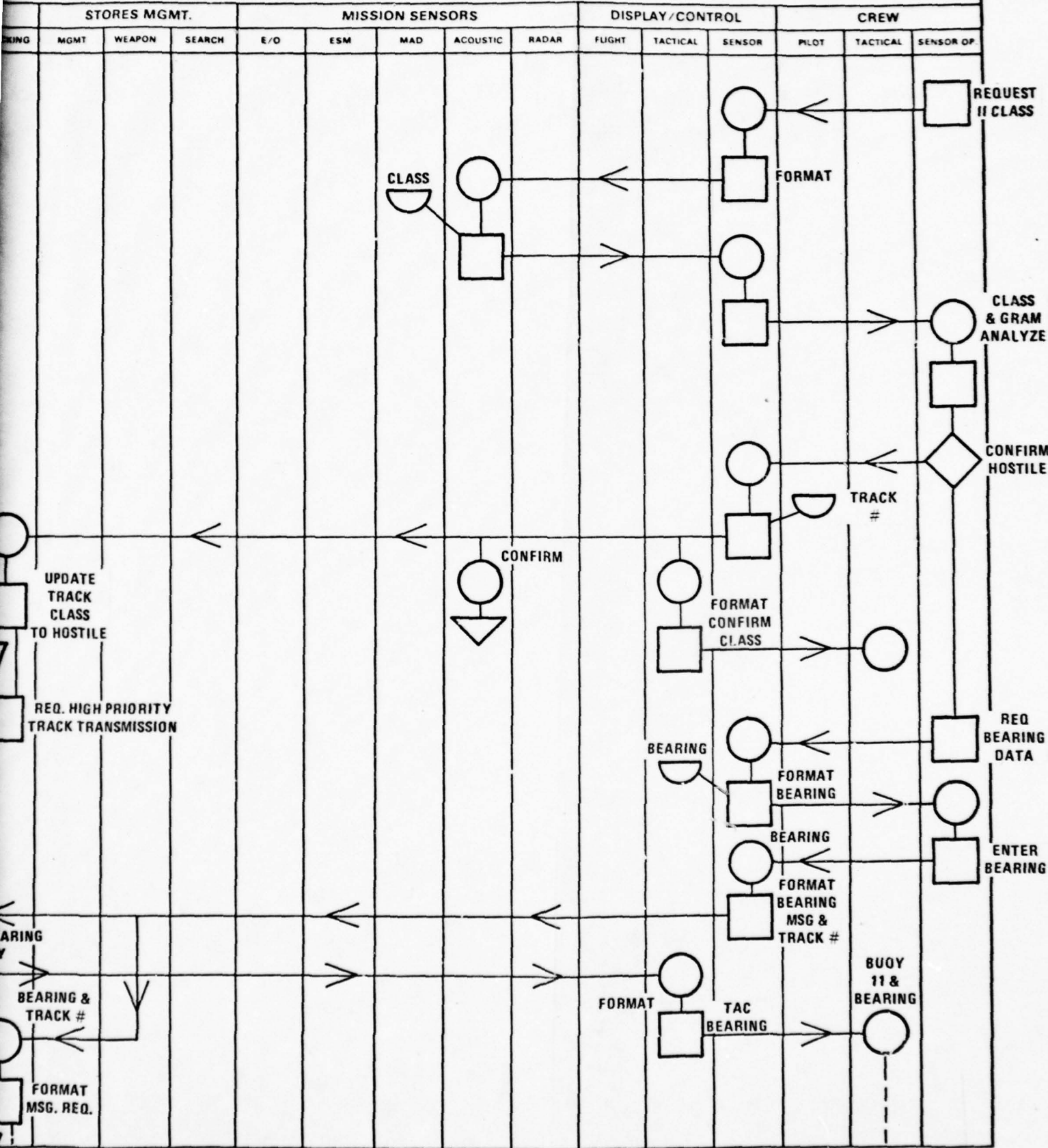


# VSTOL F3 AVIONICS CONCEPT WITH



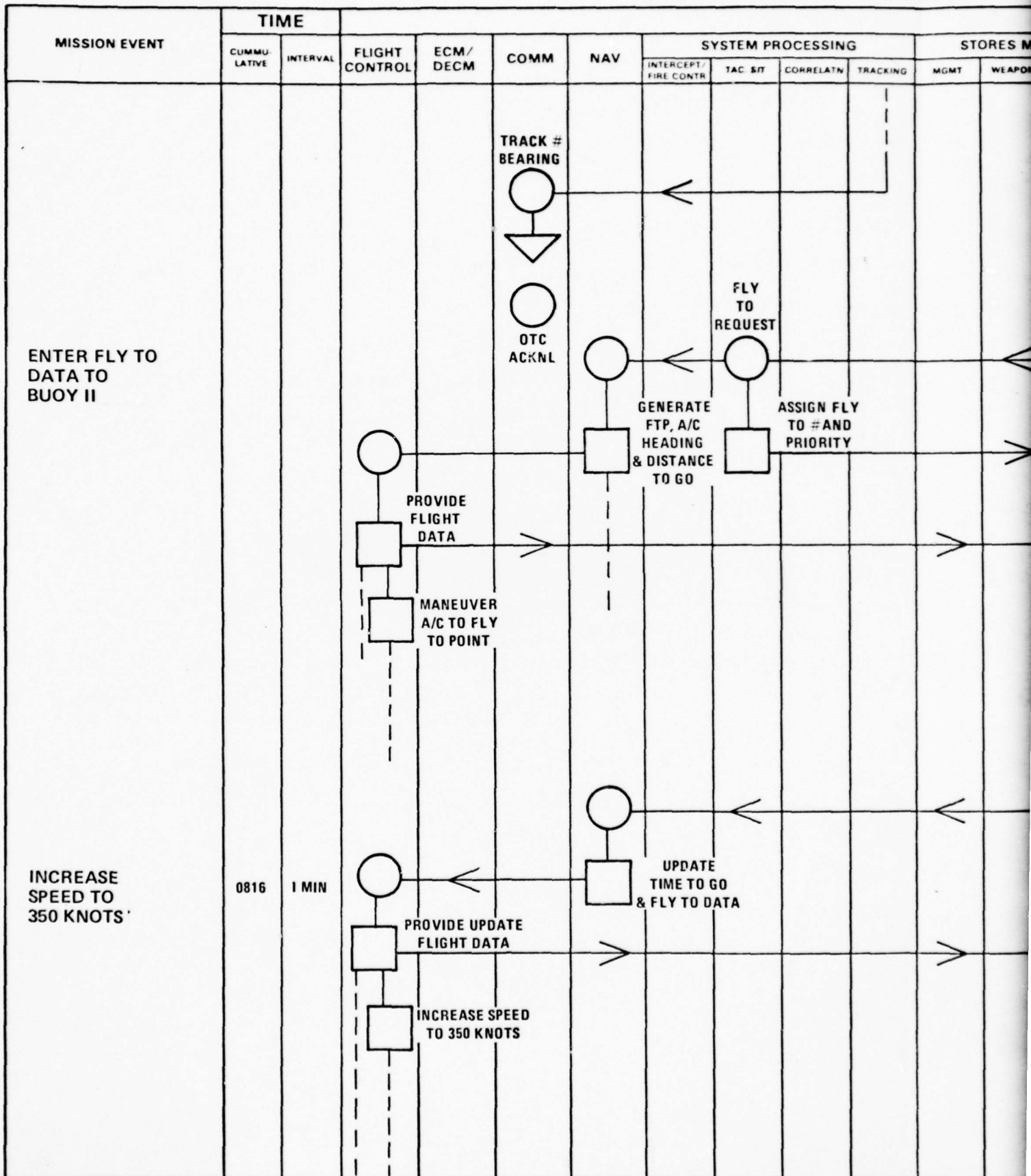
# CONCEPT WITH 3 MAN CREW

## SUBSYSTEMS



2

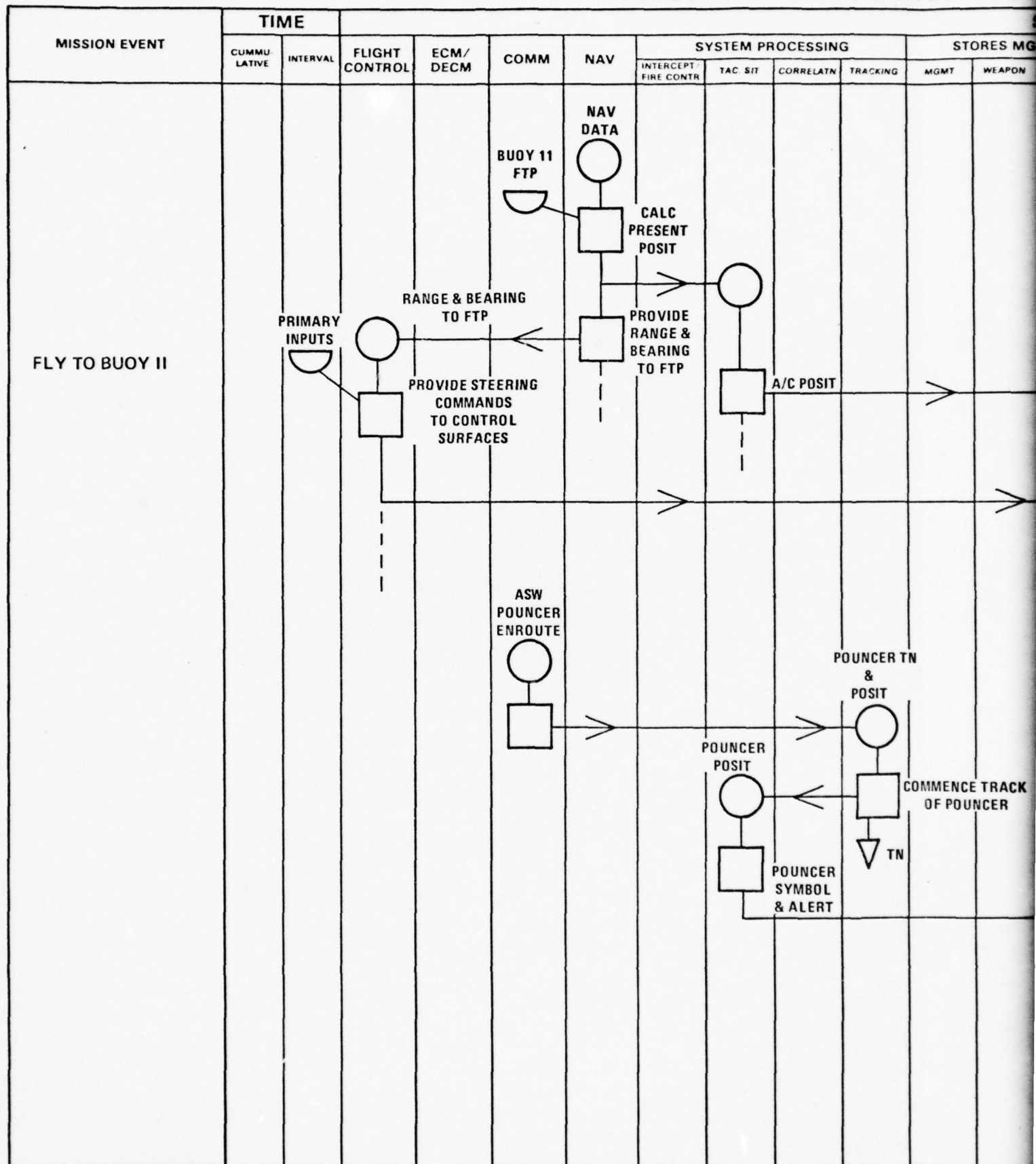
## VSTOL F3 AVIONICS CONCEPT WI



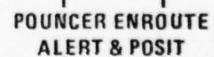


## SUBSYSTEMS





## SUBSYSTEMS



## VSTOL F3 AVIONICS CONCEPT WITH

MISSION EVENT	TIME		FLIGHT CONTROL	ECM/DECM	COMM	NAV	SYSTEM PROCESSING				STORES MGMT	
	CUMMULATIVE	INTERVAL					INTERCEPT/FIRE CONTR	TAC SIT	CORRELATN	TRACKING	MGMT	WEAPON
DETECT CRUISE MISSILE	0819											
TRANSMIT HIGH PRIORITY ALERT												

DETECT CRUISE MISSILE

0819

TRANSMIT HIGH PRIORITY ALERT

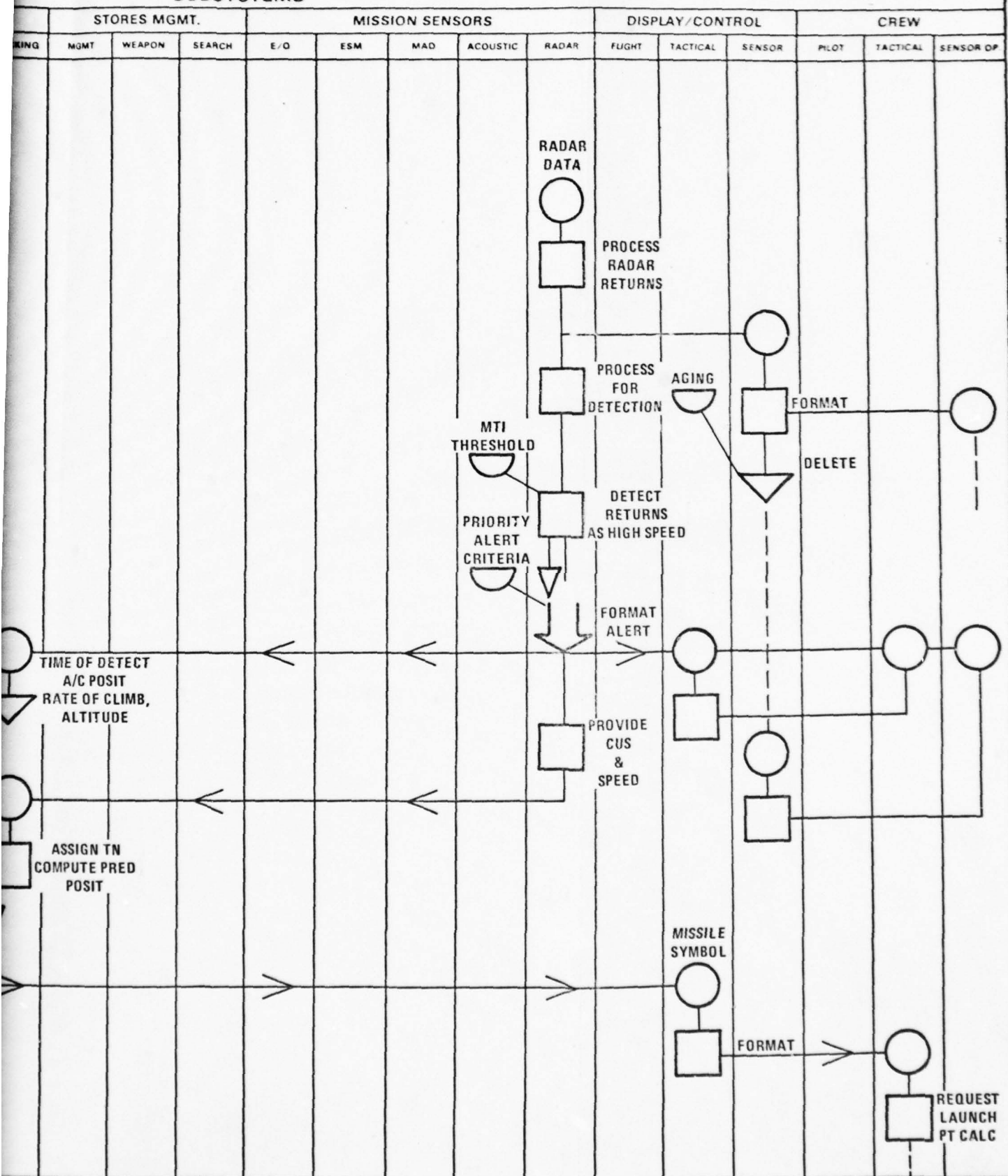
```

graph TD
    A["A/C PRESENT POSIT"] --> B(( ))
    B --> C[HIGH PRIORITY MISSILE ALERT]
    C --> D(( ))
    D --> E[PRESENT POSIT & TN]
    E --> F(( ))
    F --> G[TRANSMIT PRESENT POS T & TN]
    G -.-> H[ ]
    I["TIME OF DETECT A/C POSIT RATE OF CLIMB, ALTITUDE"] --> J(( ))
    J --> K[ASSIGN TN COMPUTE PRED POSIT]
    K --> L(( ))
    L --> M[FORMAT SYMBOL & PRED POSIT]
    M --> N[ ]
    N --> O[ ]
    O --> P[ ]
    P --> Q[ ]
    Q --> R[ ]
    R --> S[ ]
    S --> T[ ]
    T --> U[ ]
    U --> V[ ]
    V --> W[ ]
    W --> X[ ]
    X --> Y[ ]
    Y --> Z[ ]
    Z --> AA[ ]
    AA --> AB[ ]
    AB --> AC[ ]
    AC --> AD[ ]
    AD --> AE[ ]
    AE --> AF[ ]
    AF --> AG[ ]
    AG --> AH[ ]
    AH --> AI[ ]
    AI --> AJ[ ]
    AJ --> AK[ ]
    AK --> AL[ ]
    AL --> AM[ ]
    AM --> AN[ ]
    AN --> AO[ ]
    AO --> AP[ ]
    AP --> AQ[ ]
    AQ --> AR[ ]
    AR --> AS[ ]
    AS --> AT[ ]
    AT --> AU[ ]
    AU --> AV[ ]
    AV --> AW[ ]
    AW --> AX[ ]
    AX --> AY[ ]
    AY --> AZ[ ]
    AZ --> BA[ ]
    BA --> BB[ ]
    BB --> BC[ ]
    BC --> BD[ ]
    BD --> BE[ ]
    BE --> BF[ ]
    BF --> BG[ ]
    BG --> BH[ ]
    BH --> BI[ ]
    BI --> BJ[ ]
    BJ --> BK[ ]
    BK --> BL[ ]
    BL --> BM[ ]
    BM --> BN[ ]
    BN --> BO[ ]
    BO --> BP[ ]
    BP --> BQ[ ]
    BQ --> BR[ ]
    BR --> BS[ ]
    BS --> BT[ ]
    BT --> BU[ ]
    BU --> BV[ ]
    BV --> BW[ ]
    BW --> BX[ ]
    BX --> BY[ ]
    BY --> BZ[ ]
    BZ --> CA[ ]
    CA --> CB[ ]
    CB --> CC[ ]
    CC --> CD[ ]
    CD --> CE[ ]
    CE --> CF[ ]
    CF --> CG[ ]
    CG --> CH[ ]
    CH --> CI[ ]
    CI --> CJ[ ]
    CJ --> CK[ ]
    CK --> CL[ ]
    CL --> CM[ ]
    CM --> CN[ ]
    CN --> CO[ ]
    CO --> CP[ ]
    CP --> CQ[ ]
    CQ --> CR[ ]
    CR --> CS[ ]
    CS --> CT[ ]
    CT --> CU[ ]
    CU --> CV[ ]
    CV --> CW[ ]
    CW --> CX[ ]
    CX --> CY[ ]
    CY --> CZ[ ]
    CZ --> DA[ ]
    DA --> DB[ ]
    DB --> DC[ ]
    DC --> DD[ ]
    DD --> DE[ ]
    DE --> DF[ ]
    DF --> DG[ ]
    DG --> DH[ ]
    DH --> DI[ ]
    DI --> DJ[ ]
    DJ --> DK[ ]
    DK --> DL[ ]
    DL --> DM[ ]
    DM --> DN[ ]
    DN --> DO[ ]
    DO --> DP[ ]
    DP --> DQ[ ]
    DQ --> DR[ ]
    DR --> DS[ ]
    DS --> DT[ ]
    DT --> DU[ ]
    DU --> DV[ ]
    DV --> DW[ ]
    DW --> DX[ ]
    DX --> DY[ ]
    DY --> DZ[ ]
    DZ --> EA[ ]
    EA --> EB[ ]
    EB --> EC[ ]
    EC --> ED[ ]
    ED --> EE[ ]
    EE --> EF[ ]
    EF --> EG[ ]
    EG --> EH[ ]
    EH --> EI[ ]
    EI --> EJ[ ]
    EJ --> EK[ ]
    EK --> EL[ ]
    EL --> EM[ ]
    EM --> EN[ ]
    EN --> EO[ ]
    EO --> EP[ ]
    EP --> EQ[ ]
    EQ --> ER[ ]
    ER --> ES[ ]
    ES --> ET[ ]
    ET --> EU[ ]
    EU --> EV[ ]
    EV --> EW[ ]
    EW --> EX[ ]
    EX --> EY[ ]
    EY --> EZ[ ]
    EZ --> FA[ ]
    FA --> FB[ ]
    FB --> FC[ ]
    FC --> FD[ ]
    FD --> FE[ ]
    FE --> FF[ ]
    FF --> FG[ ]
    FG --> FH[ ]
    FH --> FI[ ]
    FI --> FJ[ ]
    FJ --> FK[ ]
    FK --> FL[ ]
    FL --> FM[ ]
    FM --> FN[ ]
    FN --> FO[ ]
    FO --> FP[ ]
    FP --> FQ[ ]
    FQ --> FR[ ]
    FR --> FS[ ]
    FS --> FT[ ]
    FT --> FU[ ]
    FU --> FV[ ]
    FV --> FW[ ]
    FW --> FX[ ]
    FX --> FY[ ]
    FY --> FZ[ ]
    FZ --> GA[ ]
    GA --> GB[ ]
    GB --> GC[ ]
    GC --> GD[ ]
    GD --> GE[ ]
    GE --> GF[ ]
    GF --> GG[ ]
    GG --> GH[ ]
    GH --> GI[ ]
    GI --> GJ[ ]
    GJ --> GK[ ]
    GK --> GL[ ]
    GL --> GM[ ]
    GM --> GN[ ]
    GN --> GO[ ]
    GO --> GP[ ]
    GP --> GQ[ ]
    GQ --> GR[ ]
    GR --> GS[ ]
    GS --> GT[ ]
    GT --> GU[ ]
    GU --> GV[ ]
    GV --> GW[ ]
    GW --> GX[ ]
    GX --> GY[ ]
    GY --> GZ[ ]
    GZ --> HA[ ]
    HA --> HB[ ]
    HB --> HC[ ]
    HC --> HD[ ]
    HD --> HE[ ]
    HE --> HF[ ]
    HF --> HG[ ]
    HG --> HH[ ]
    HH --> HI[ ]
    HI --> HJ[ ]
    HJ --> HK[ ]
    HK --> HL[ ]
    HL --> HM[ ]
    HM --> HN[ ]
    HN --> HO[ ]
    HO --> HP[ ]
    HP --> HQ[ ]
    HQ --> HR[ ]
    HR --> HS[ ]
    HS --> HT[ ]
    HT --> HU[ ]
    HU --> HV[ ]
    HV --> HW[ ]
    HW --> HX[ ]
    HX --> HY[ ]
    HY --> HZ[ ]
    HZ --> IA[ ]
    IA --> IB[ ]
    IB --> IC[ ]
    IC --> ID[ ]
    ID --> IE[ ]
    IE --> IF[ ]
    IF --> IG[ ]
    IG --> IH[ ]
    IH --> II[ ]
    II --> IJ[ ]
    IJ --> IK[ ]
    IK --> IL[ ]
    IL --> IM[ ]
    IM --> IN[ ]
    IN --> IO[ ]
    IO --> IP[ ]
    IP --> IQ[ ]
    IQ --> IR[ ]
    IR --> IS[ ]
    IS --> IT[ ]
    IT --> IU[ ]
    IU --> IV[ ]
    IV --> IW[ ]
    IW --> IX[ ]
    IX --> IY[ ]
    IY --> IZ[ ]
    IZ --> JA[ ]
    JA --> JB[ ]
    JB --> JC[ ]
    JC --> JD[ ]
    JD --> JE[ ]
    JE --> JF[ ]
    JF --> JG[ ]
    JG --> JH[ ]
    JH --> JI[ ]
    JI --> JJ[ ]
    JJ --> JK[ ]
    JK --> JL[ ]
    JL --> JM[ ]
    JM --> JN[ ]
    JN --> JO[ ]
    JO --> JP[ ]
    JP --> JQ[ ]
    JQ --> JR[ ]
    JR --> JS[ ]
    JS --> JT[ ]
    JT --> JU[ ]
    JU --> JV[ ]
    JV --> JW[ ]
    JW --> JX[ ]
    JX --> JY[ ]
    JY --> JZ[ ]
    JZ --> KA[ ]
    KA --> KB[ ]
    KB --> KC[ ]
    KC --> KD[ ]
    KD --> KE[ ]
    KE --> KF[ ]
    KF --> KG[ ]
    KG --> KH[ ]
    KH --> KI[ ]
    KI --> KJ[ ]
    KJ --> KK[ ]
    KK --> KL[ ]
    KL --> KM[ ]
    KM --> KN[ ]
    KN --> KO[ ]
    KO --> KP[ ]
    KP --> KQ[ ]
    KQ --> KR[ ]
    KR --> KS[ ]
    KS --> KT[ ]
    KT --> KU[ ]
    KU --> KV[ ]
    KV --> KW[ ]
    KW --> KX[ ]
    KX --> KY[ ]
    KY --> KZ[ ]
    KZ --> LA[ ]
    LA --> LB[ ]
    LB --> LC[ ]
    LC --> LD[ ]
    LD --> LE[ ]
    LE --> LF[ ]
    LF --> LG[ ]
    LG --> LH[ ]
    LH --> LI[ ]
    LI --> LJ[ ]
    LJ --> LK[ ]
    LK --> LL[ ]
    LL --> LM[ ]
    LM --> LN[ ]
    LN --> LO[ ]
    LO --> LP[ ]
    LP --> LQ[ ]
    LQ --> LR[ ]
    LR --> LS[ ]
    LS --> LT[ ]
    LT --> LU[ ]
    LU --> LV[ ]
    LV --> LW[ ]
    LW --> LX[ ]
    LX --> LY[ ]
    LY --> LZ[ ]
    LZ --> MA[ ]
    MA --> MB[ ]
    MB --> MC[ ]
    MC --> MD[ ]
    MD --> ME[ ]
    ME -->
```

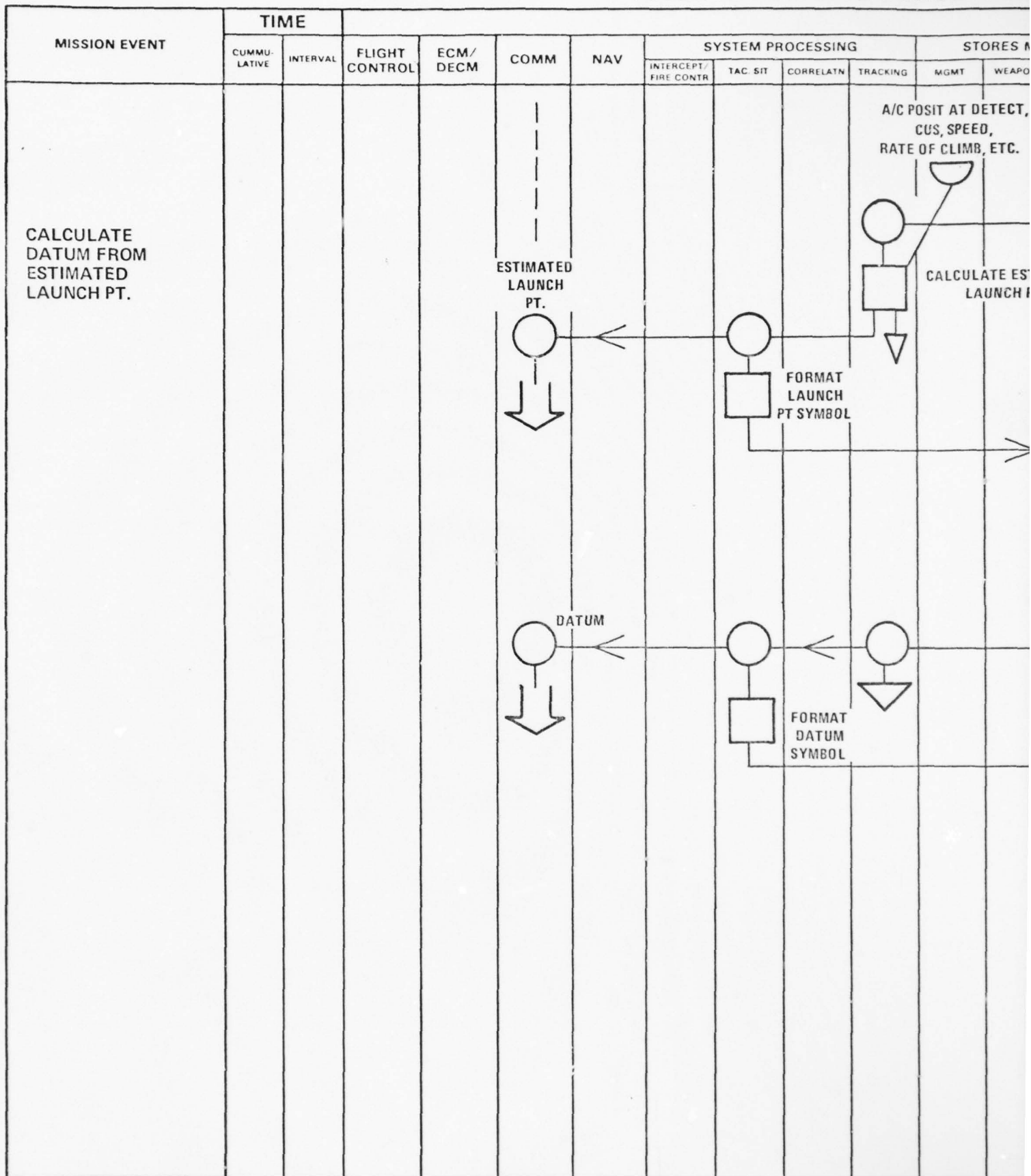


## CONCEPT WITH 3 MAN CREW

## SUBSYSTEMS

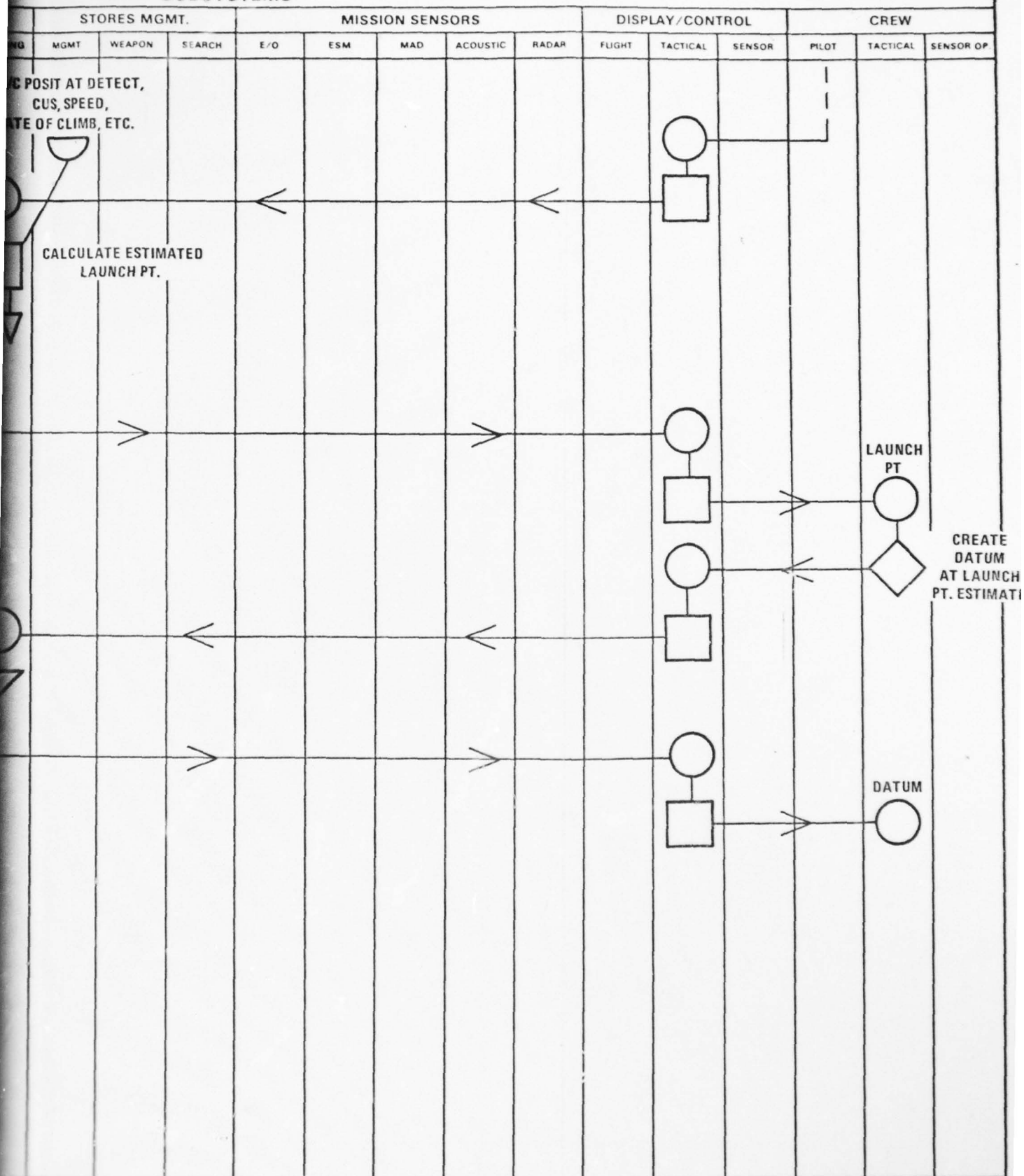


# VSTOL F3 AVIONICS CONCEPT W



CONCEPT WITH 3 MAN CREW

## SUBSYSTEMS



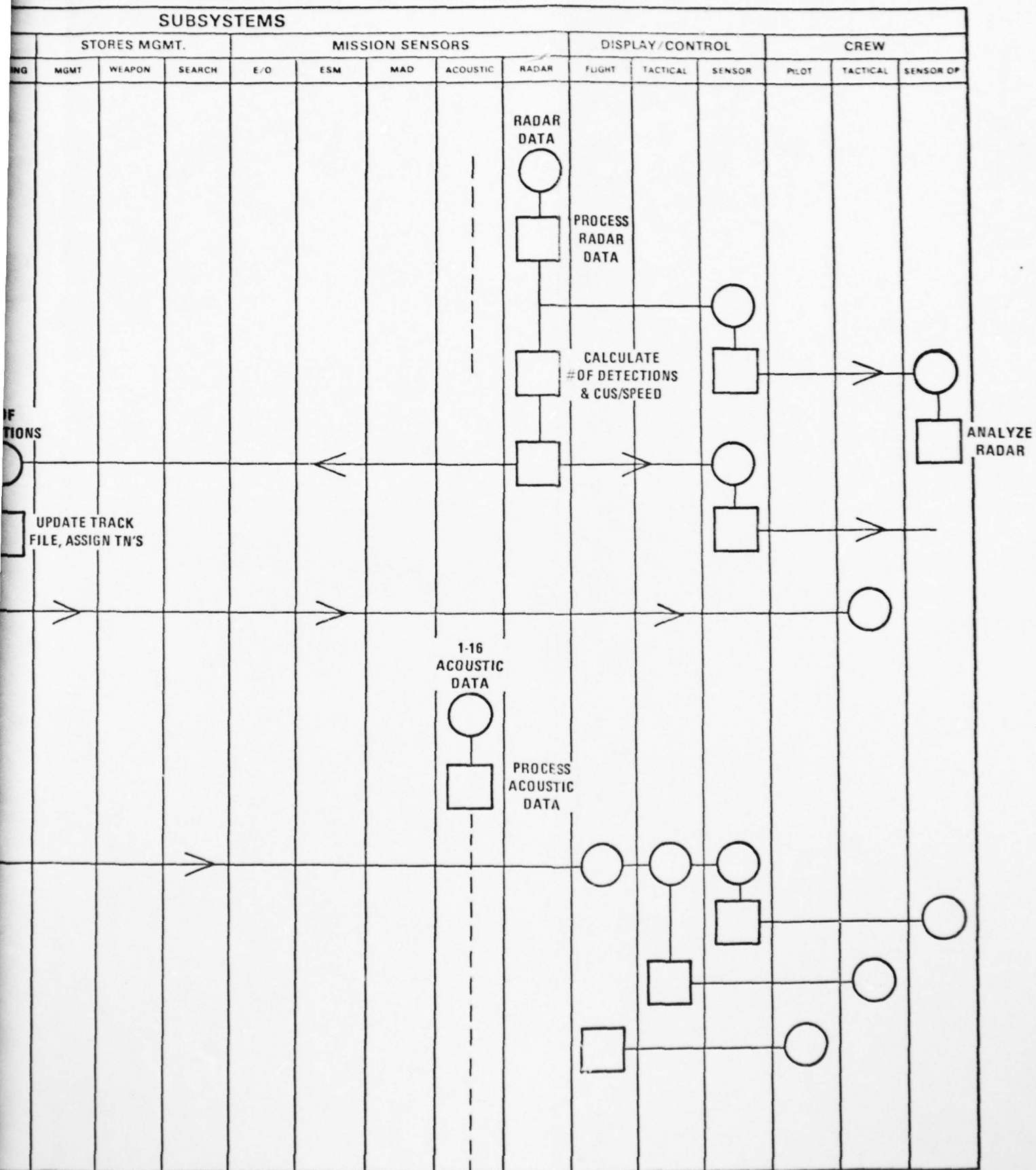
## VSTOL F3 AVIONICS CONCEPT WITH

[illegible]

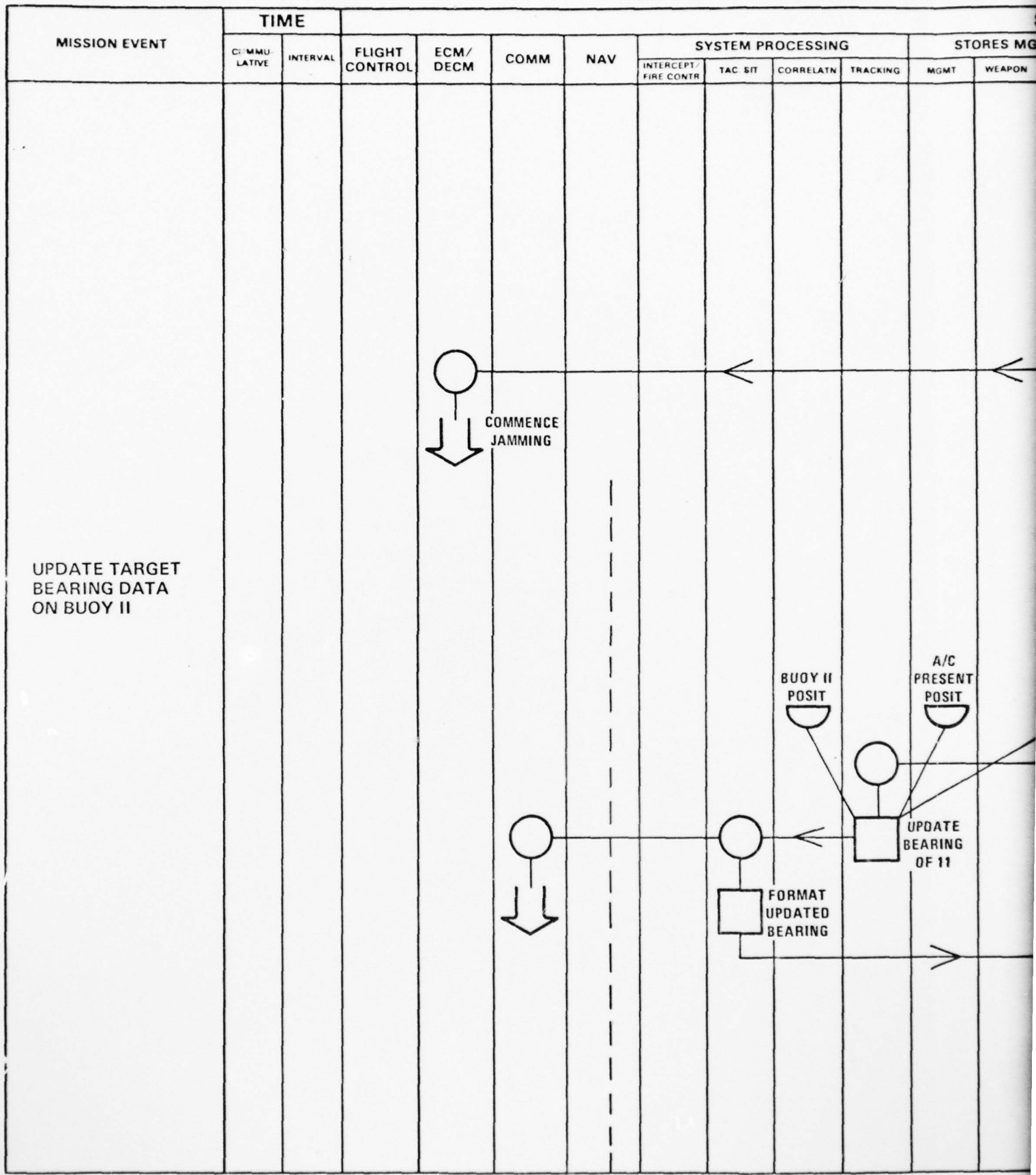


## CONCEPT WITH 3 MAN CREW

## SUBSYSTEMS



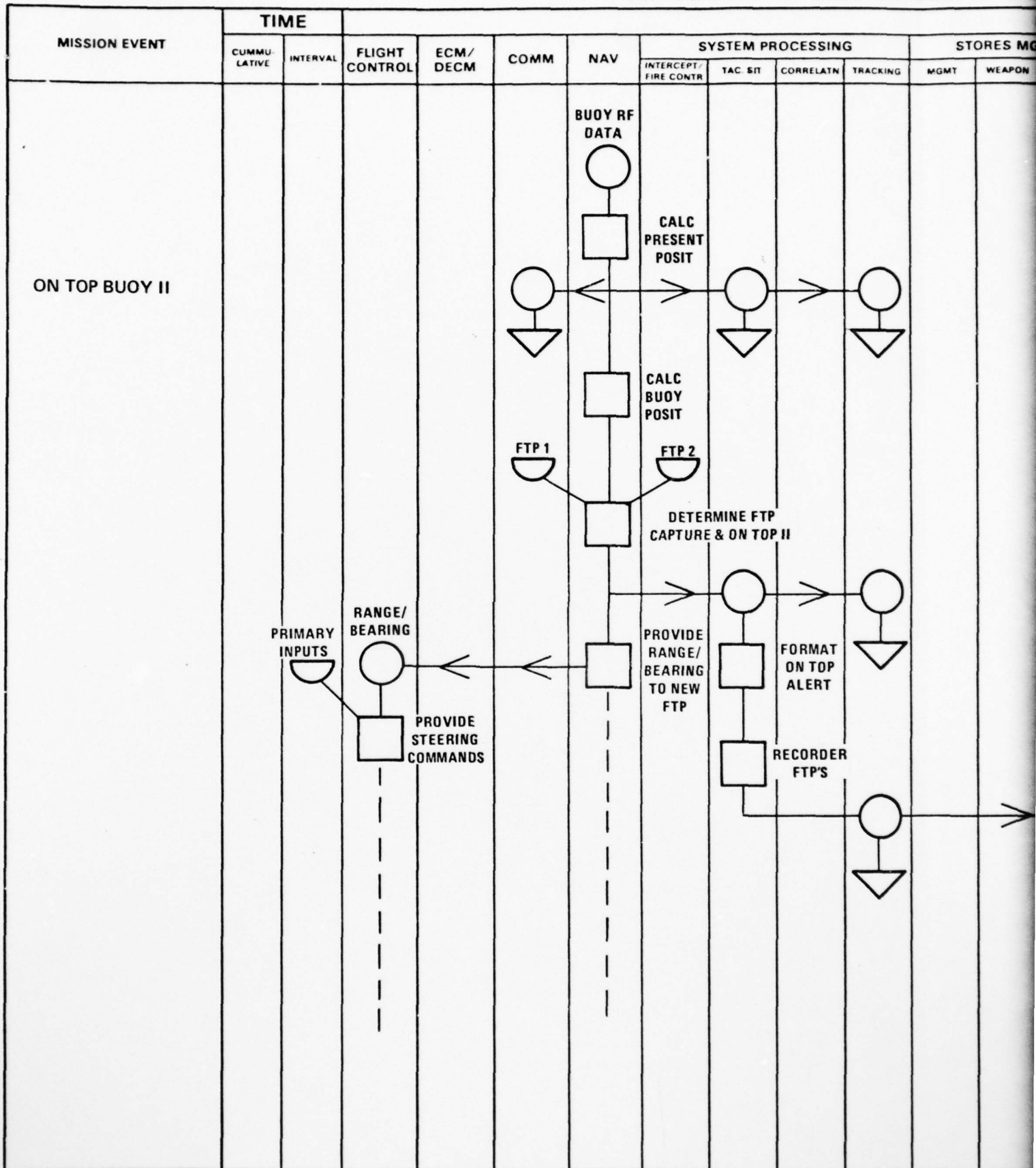
VSTOL F3 AVIONICS CONCEPT WIT



## SUBSYSTEMS



# VSTOL F3 AVIONICS CONCEPT WITH





## SUBSYSTEMS

2

## VSTOL F3 AVIONICS CONCEPT WITH

MISSION EVENT	TIME		FLIGHT CONTROL	ECM/DECM	COMM	NAV	SYSTEM PROCESSING				STORES MGMT	
	CUMMULATIVE	INTERVAL					INTERCEPT/FIRE CONTR	TAC SIT	CORRELATN	TRACKING	MGMT	WEAPON
DETECTORS OF DATA ON II												
NOTIFY OTC OF TARGET LOSS												
MONITOR ADJACENT BUOYS												

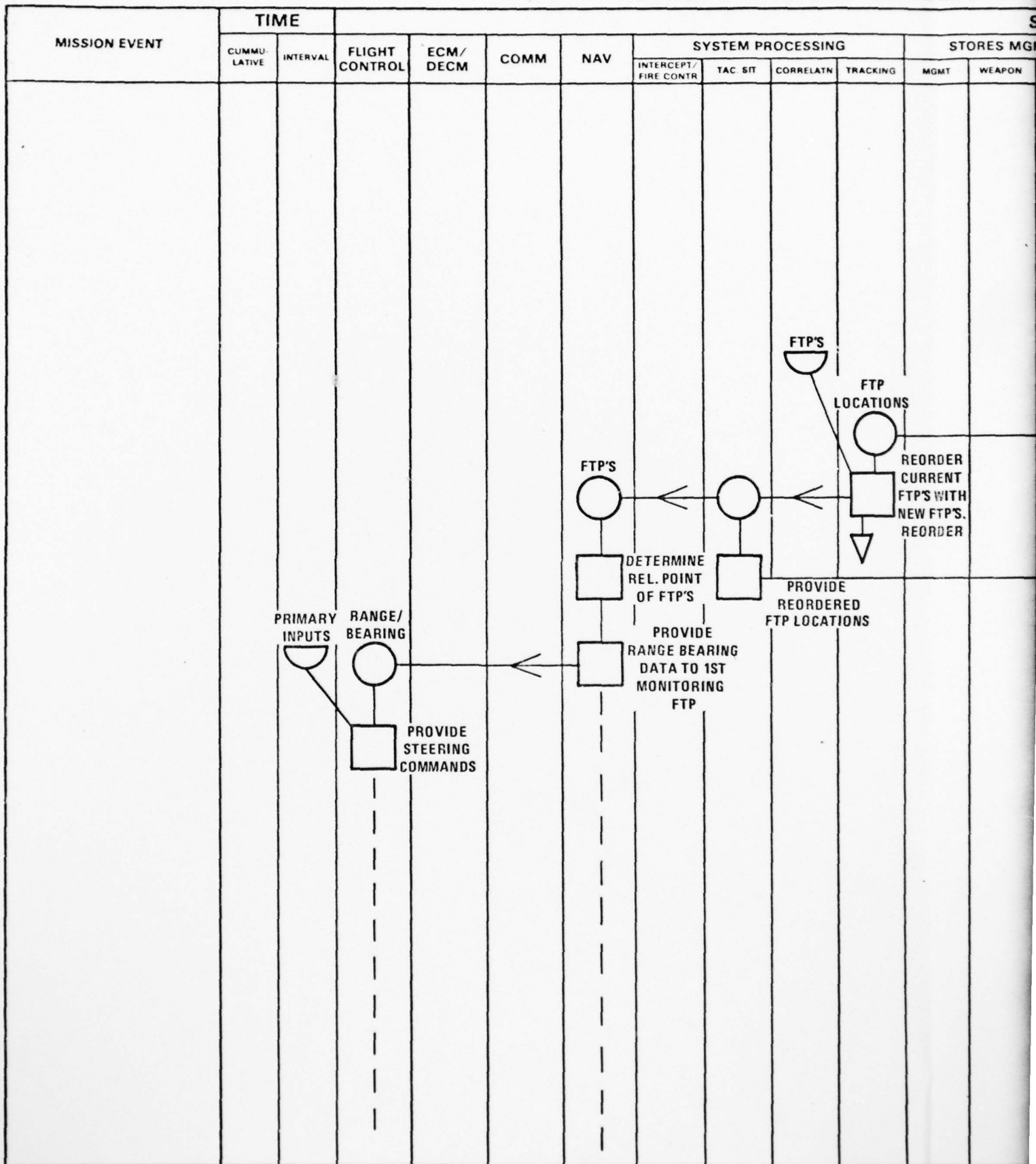
```

graph TD
    A((LOST CONTACT ALERT, BEARING & CUS.)) --> B((FORMAT ALERT, UPDATE SYMBOLOGY))
    B --> C[FORMAT LAST CONTACT ALERT UPDATE TN'S, PROVIDE]
    C --> D(( ))
  
```

## SUBSYSTEMS



# VSTOL F3 AVIONICS CONCEPT WITH





CEPT WITH 3 MAN CREW

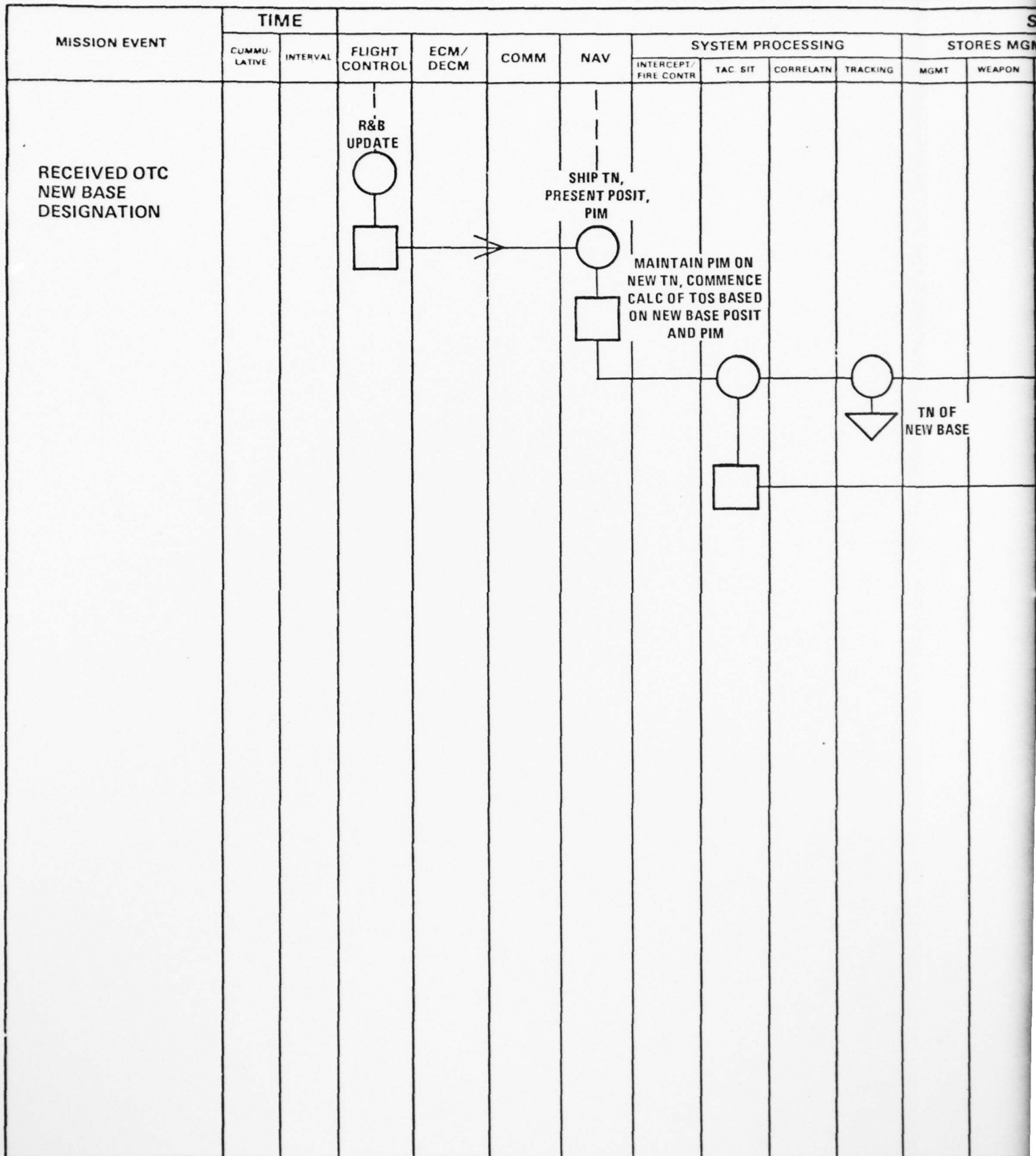
## SUBSYSTEMS

STORES MGMT.			MISSION SENSORS					DISPLAY/CONTROL			CREW		
QNT	WEAPON	SEARCH	E/O	ESM	MAD	ACOUSTIC	RADAR	FLIGHT	TACTICAL	SENSOR	PILOT	TACTICAL	SENSOR OP.

INSERT FTP's  
FOR MONITORING  
ORBIT OF ADJ.  
BUOYS

## FTP's

# VSTOL F3 AVIONICS CONCEPT WITH

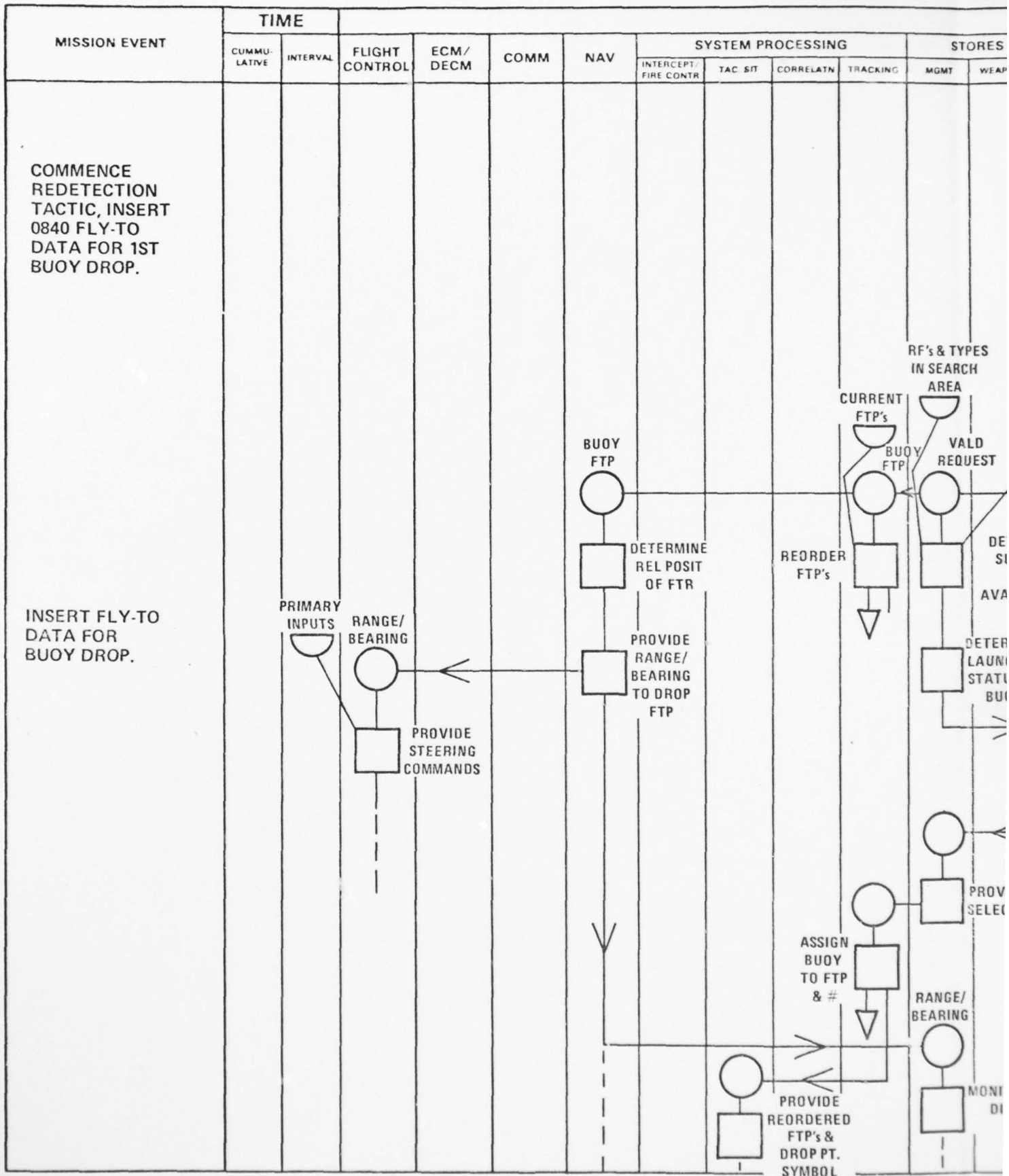


INCEPT WITH 3 MAN CREW

## SUBSYSTEMS

STORES MGMT.			MISSION SENSORS					DISPLAY/CONTROL			CREW		
MGMT	WEAPON	SEARCH	E/O	ESM	MAD	ACOUSTIC	RADAR	FLIGHT	TACTICAL	SENSOR	PILOT	TACTICAL	SENSOR OP.
										</			

# VSTOL F3 AVIONICS CONCEPT W





AD-A072 427

NAVAL AIR DEVELOPMENT CENTER WARMINSTER PA COMMUNICA--ETC F/G 5/2  
BASIC LABORATORY ARCHITECTURE PLAN.(U)  
MAY 79 S GREENBERG, C JOECKEL, R MEJZAK

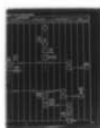
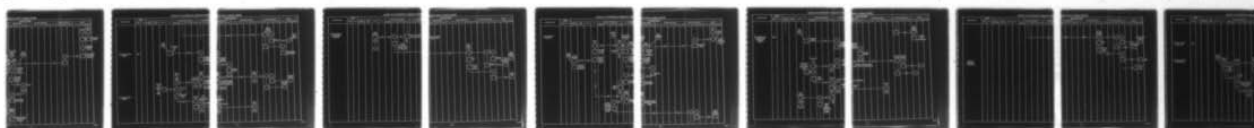
IDWA-45-78-04

UNCLASSIFIED

NADC-79161-40

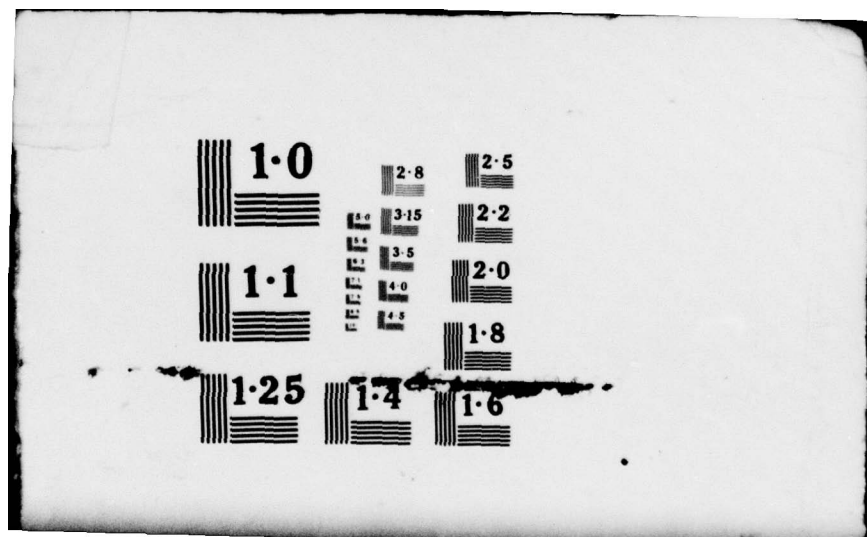
NL

4 OF 4  
AD  
A072427



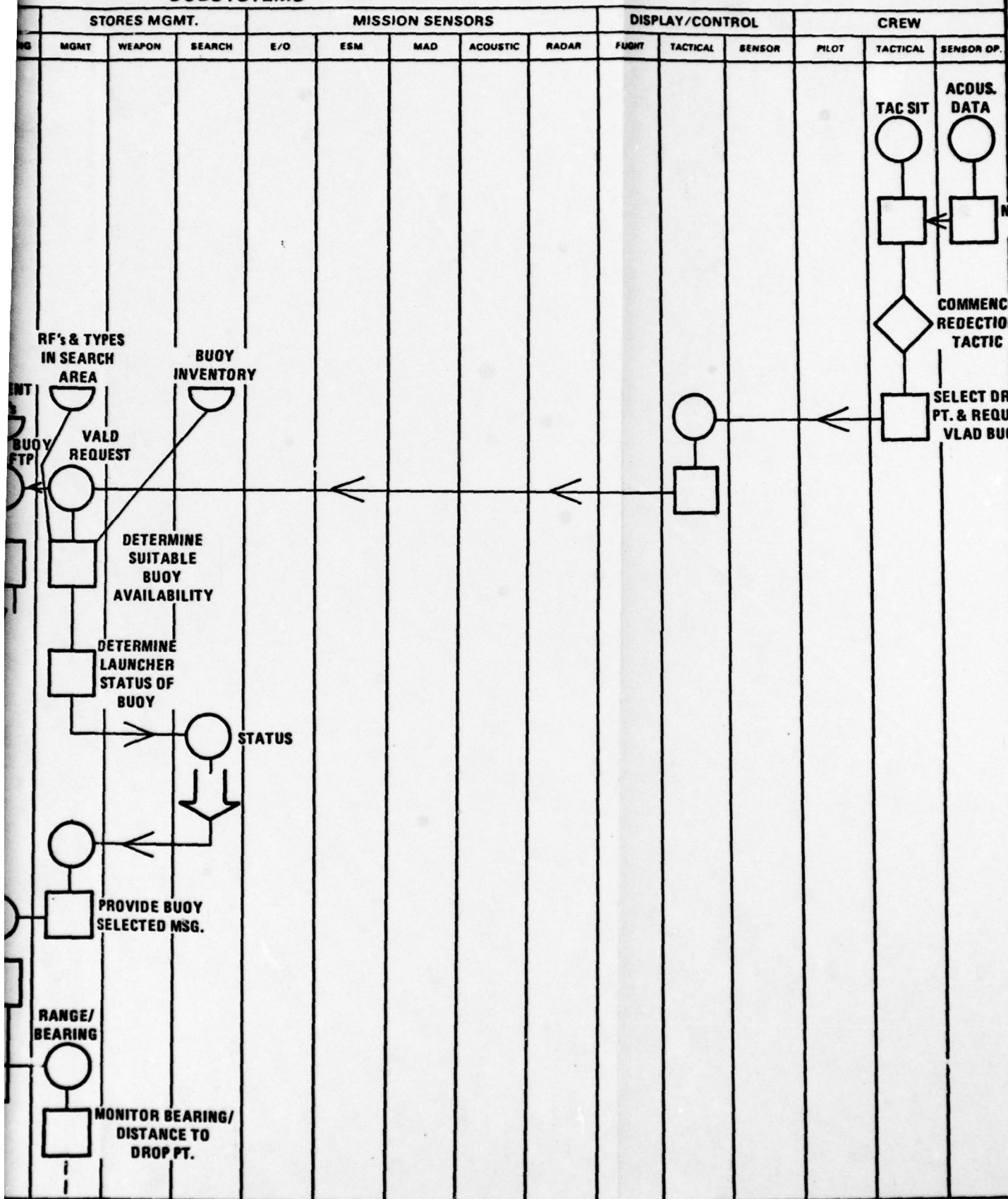
END  
DATE  
FILMED

9 - 79  
DDC

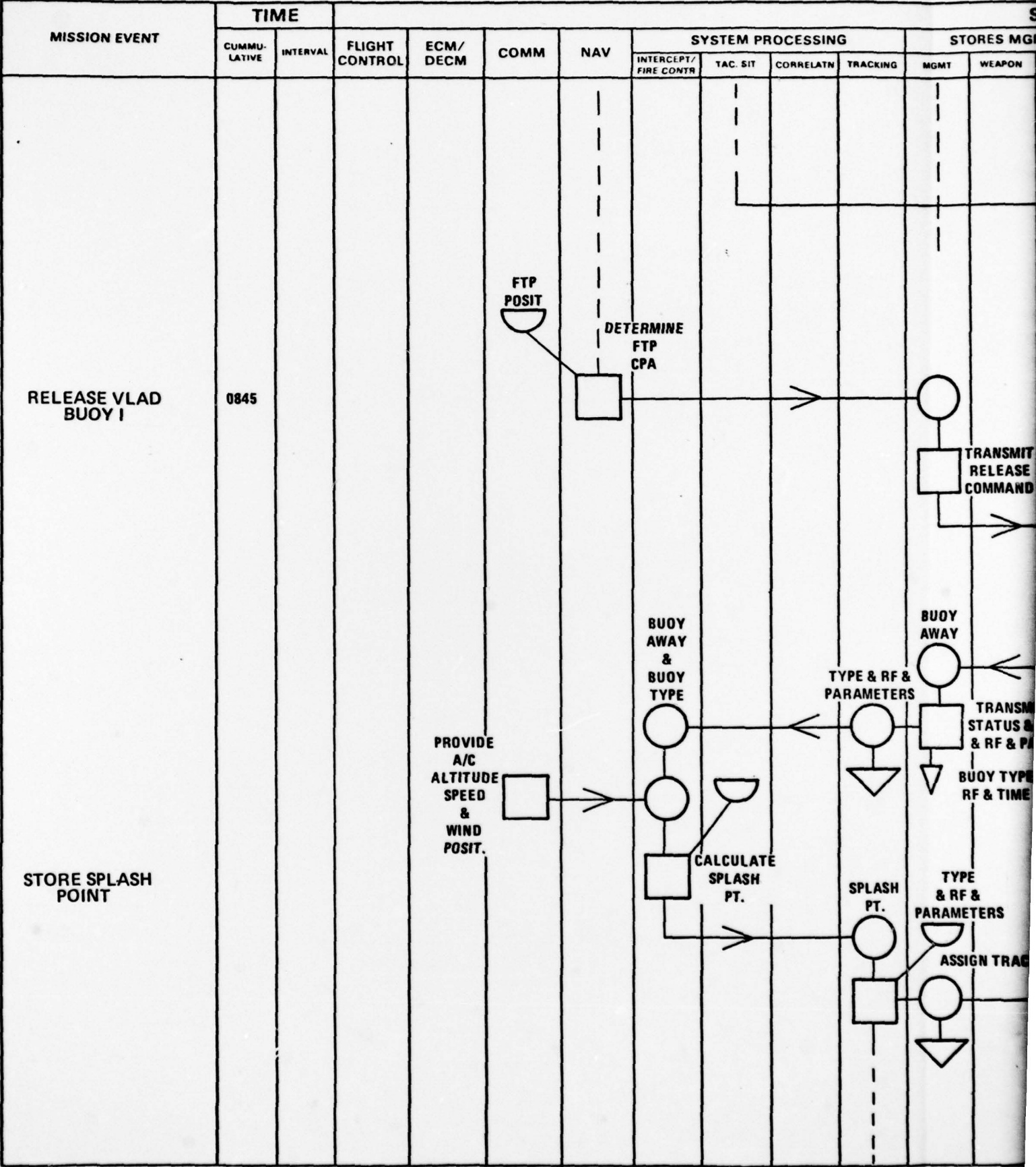


# CONCEPT WITH 3 MAN CREW

## SUBSYSTEMS



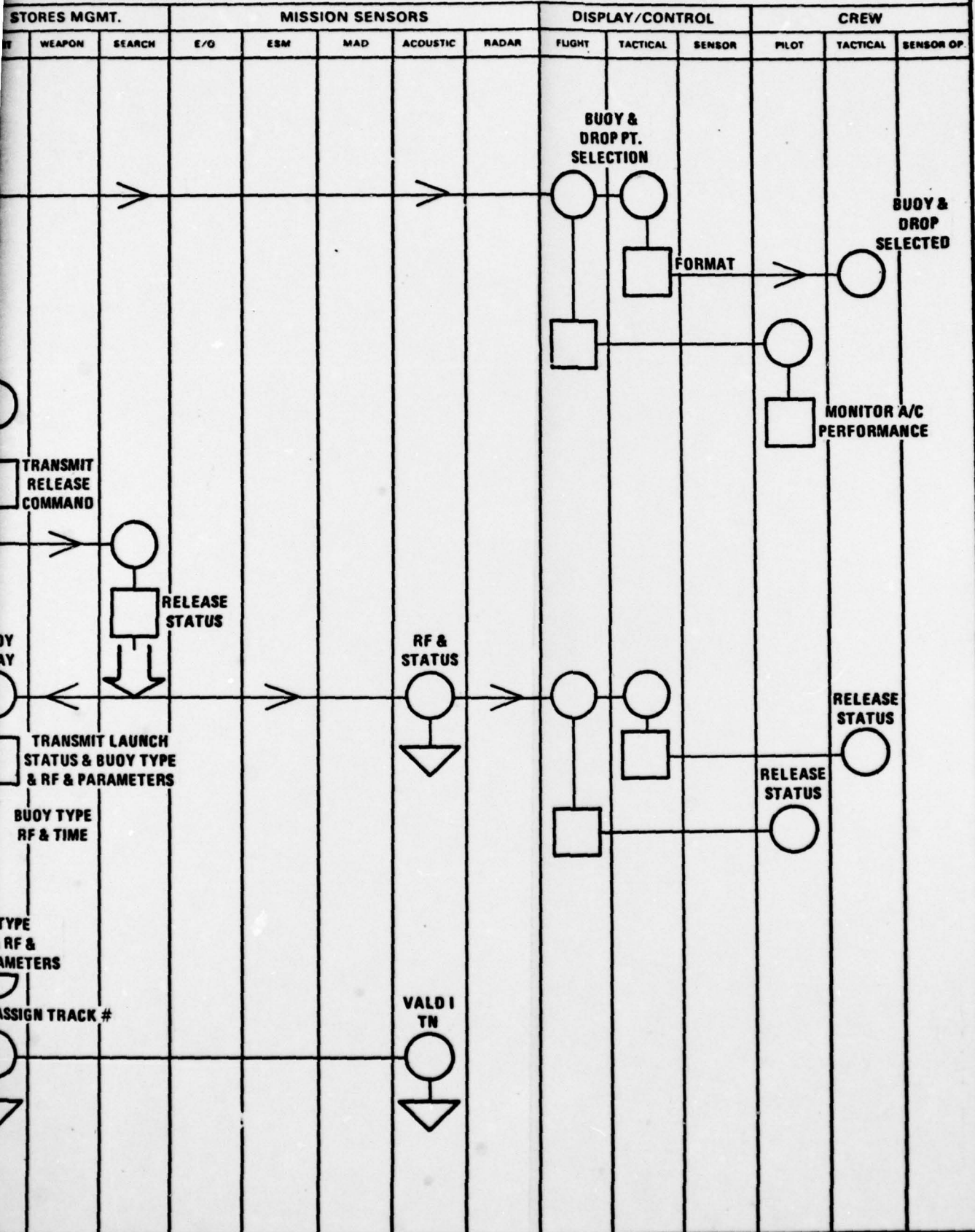
VSTOL F3 AVIONICS CONCEPT WITH



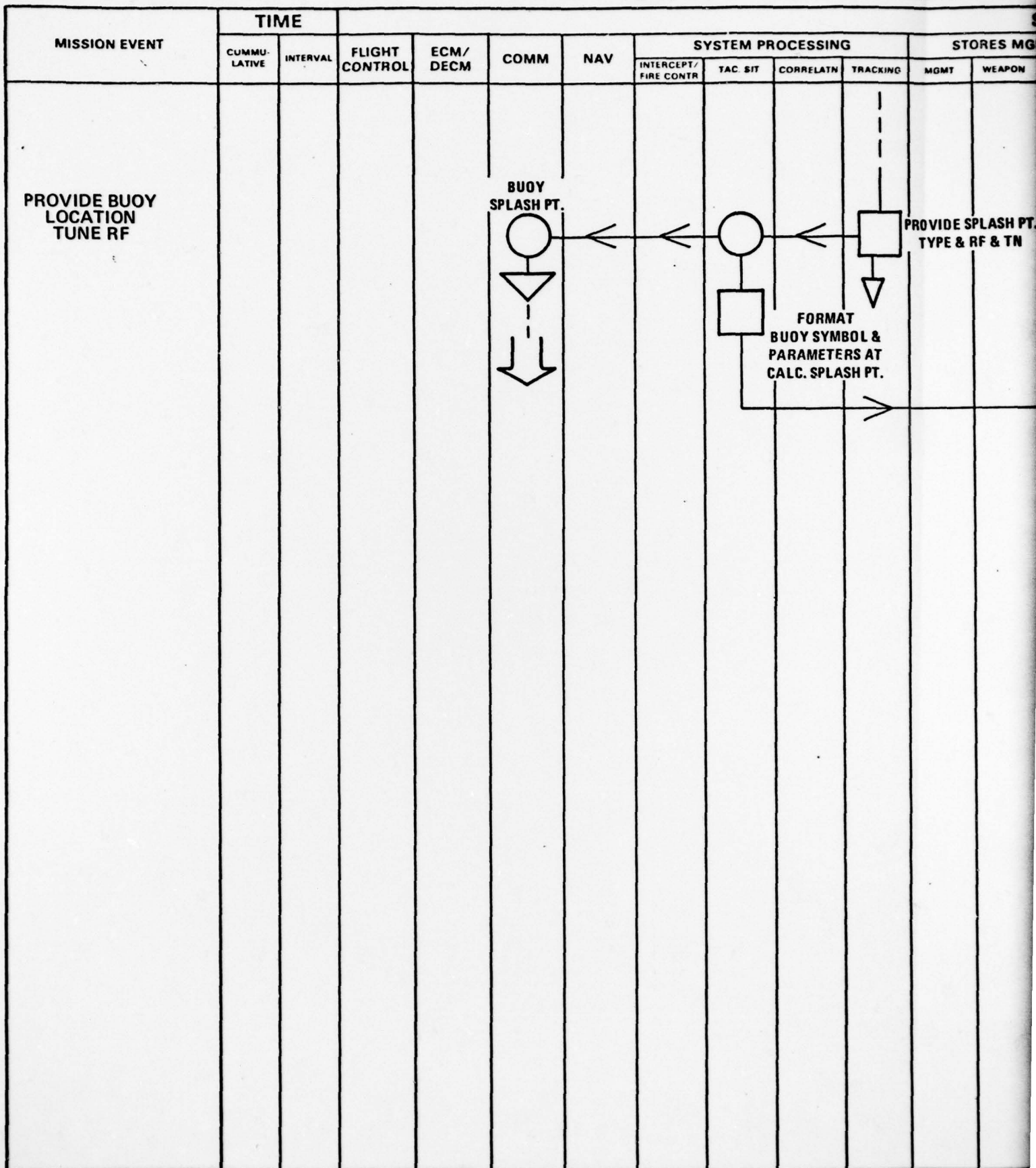


# CEPT WITH 3 MAN CREW

## SUBSYSTEMS

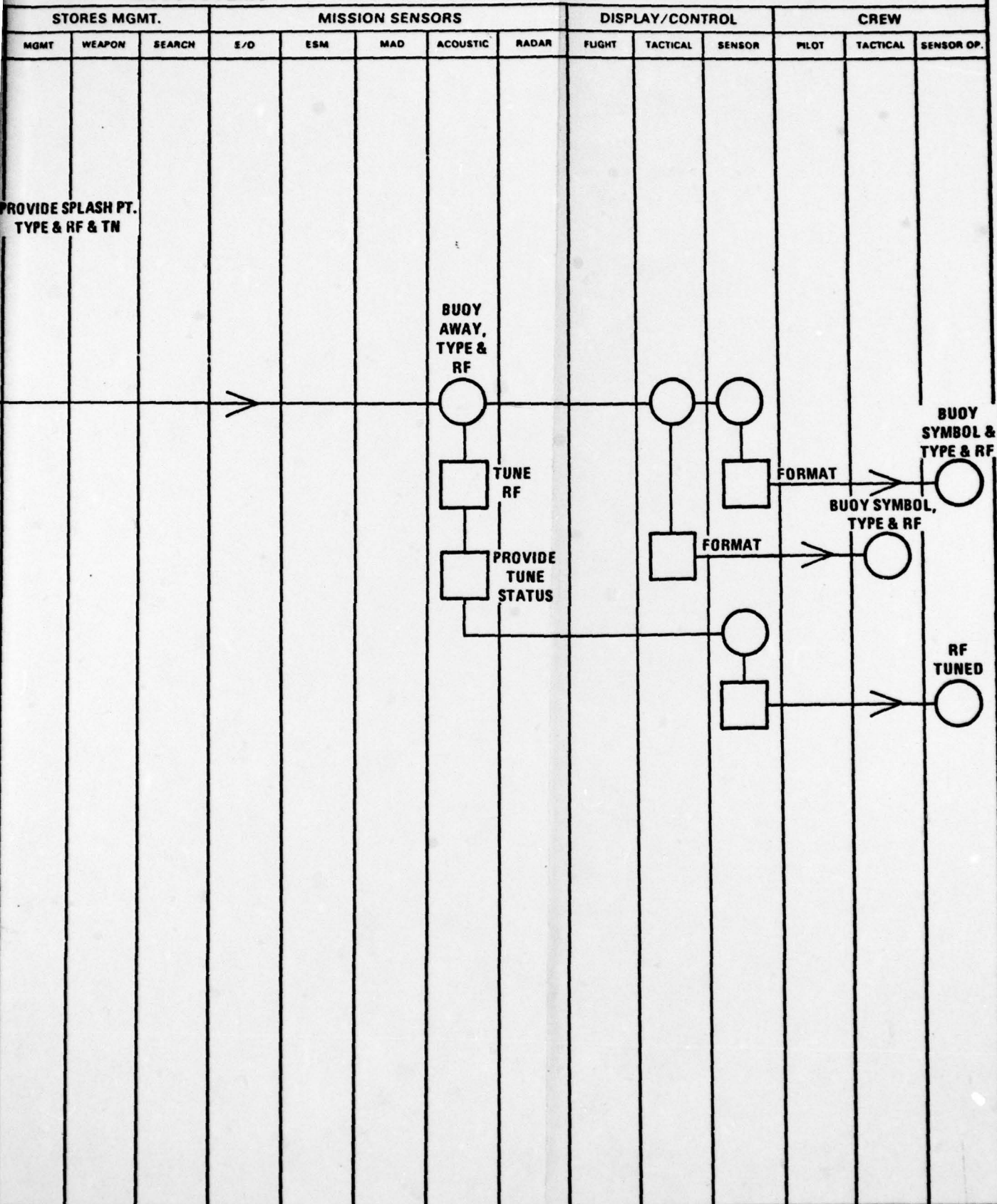


# VSTOL F3 AVIONICS CONCEPT WITH



# CONCEPT WITH 3 MAN CREW

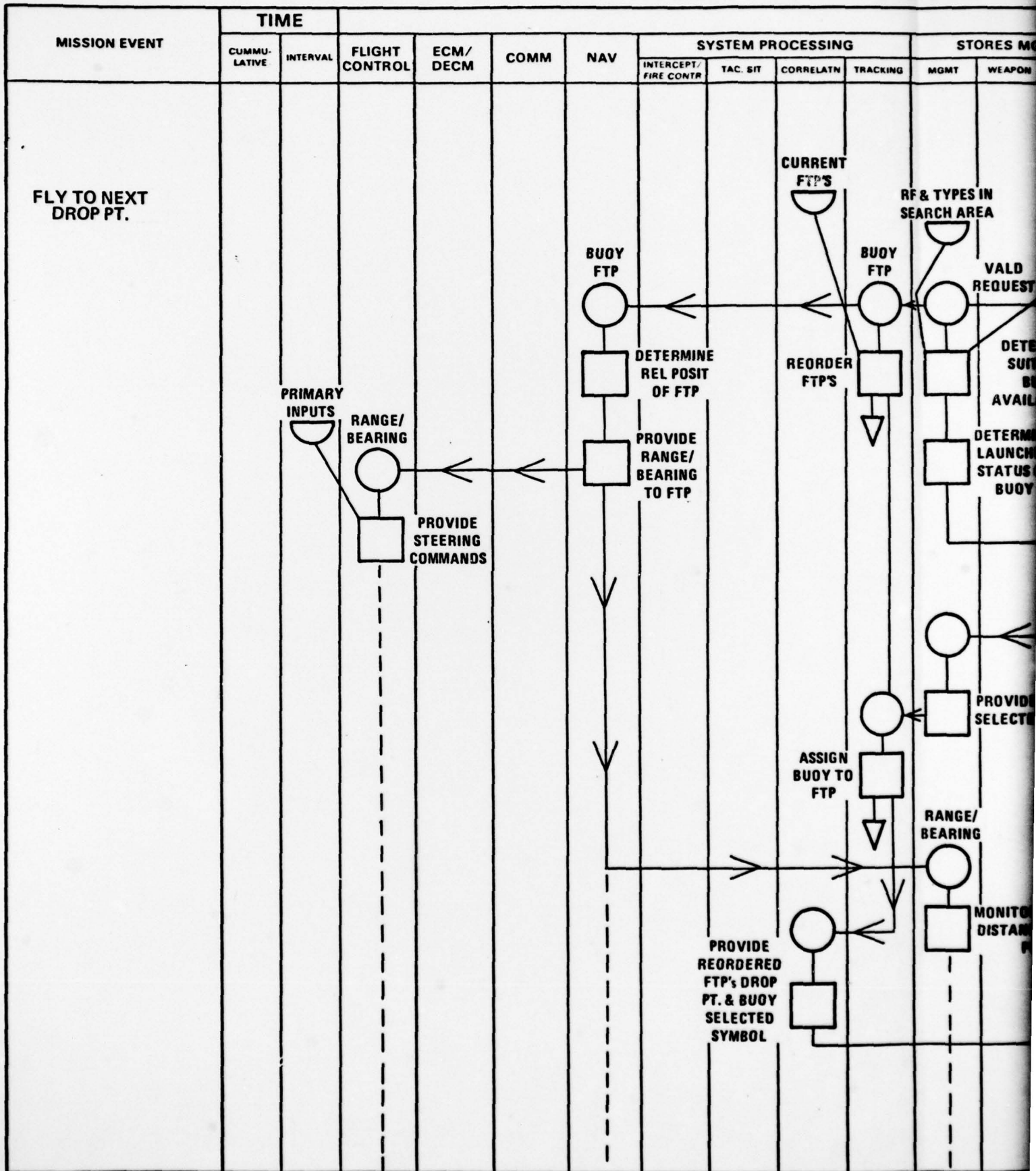
## SUBSYSTEMS



2



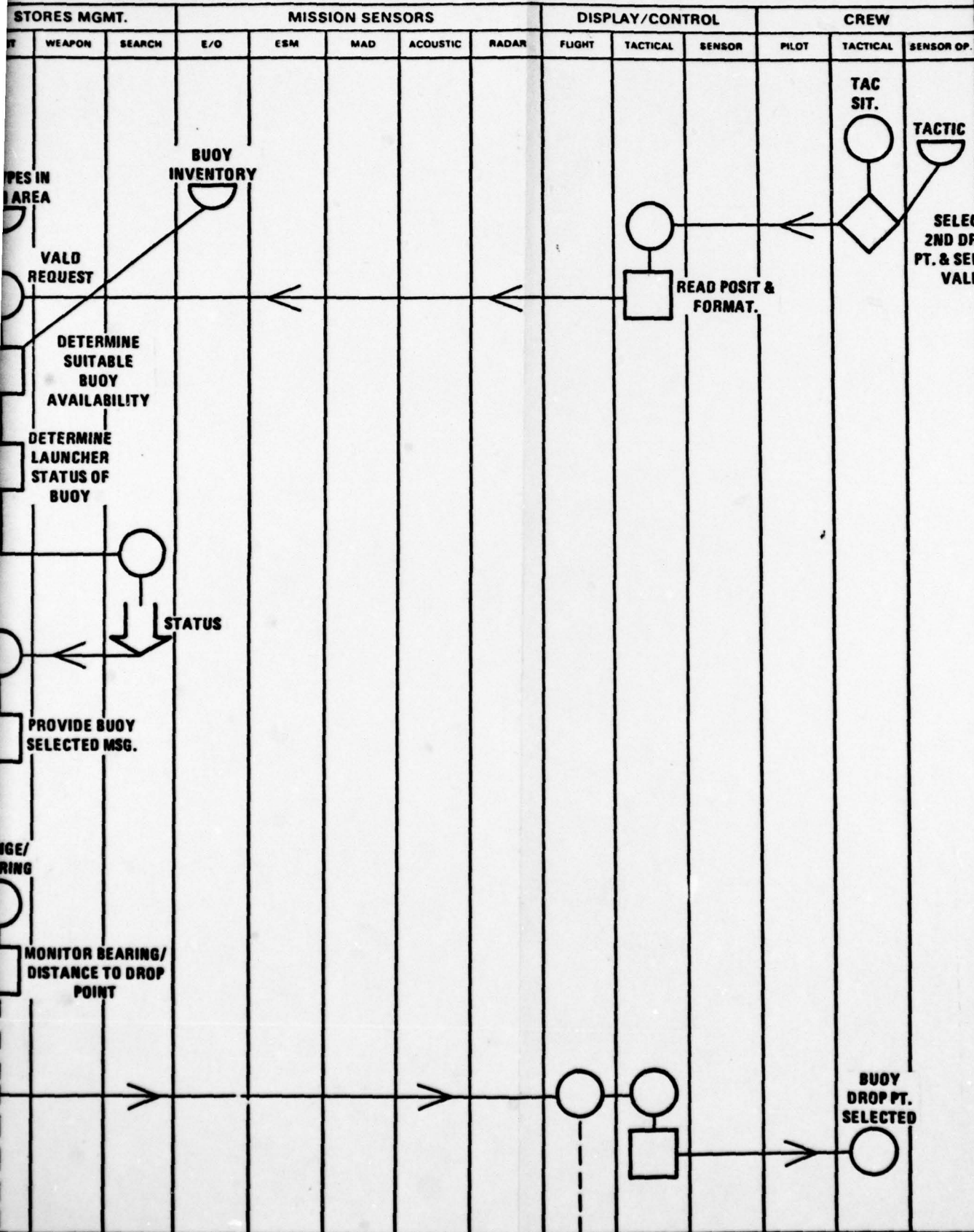
# VSTOL F3 AVIONICS CONCEPT WITH



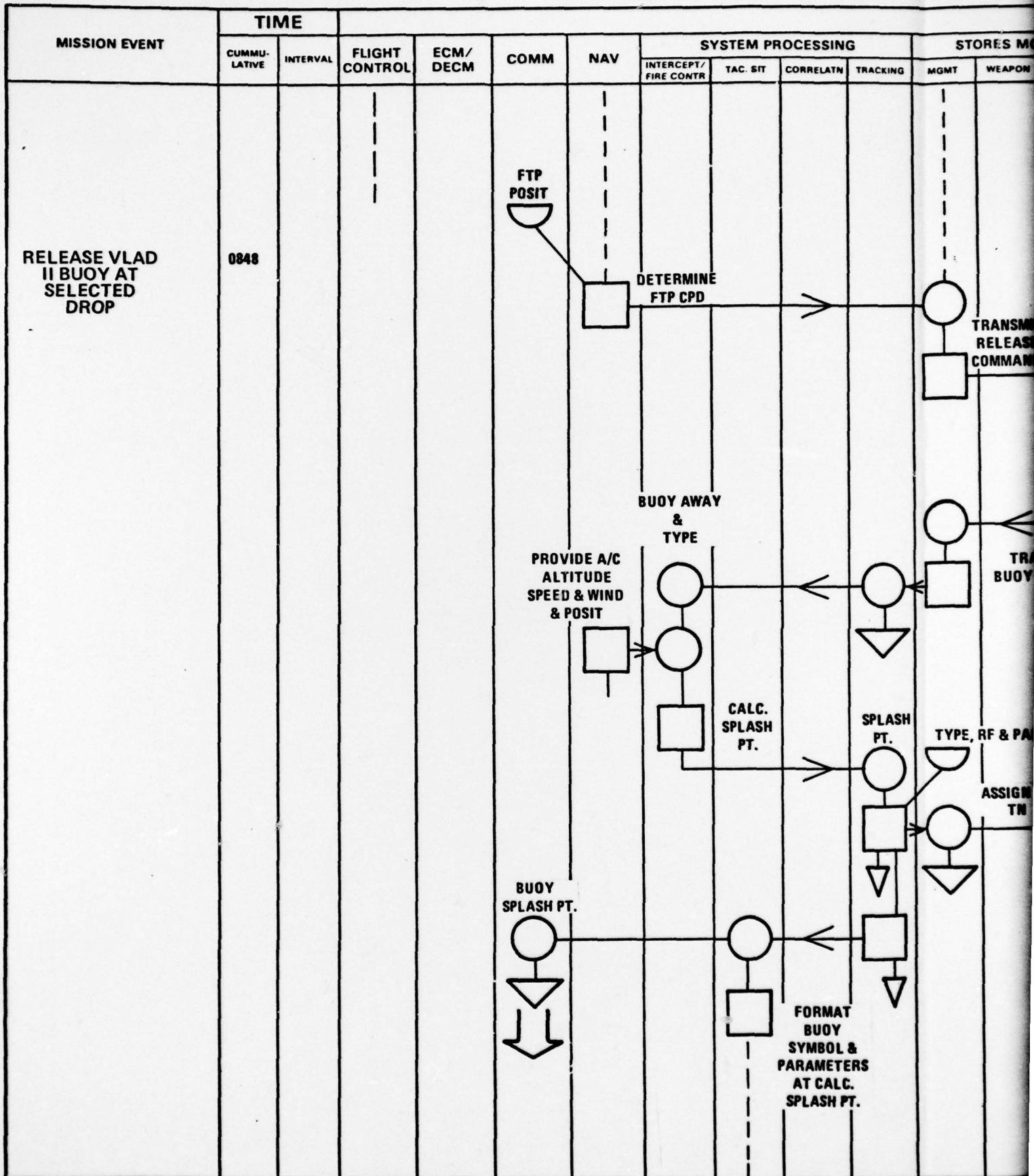


# CEPT WITH 3 MAN CREW

## SUBSYSTEMS

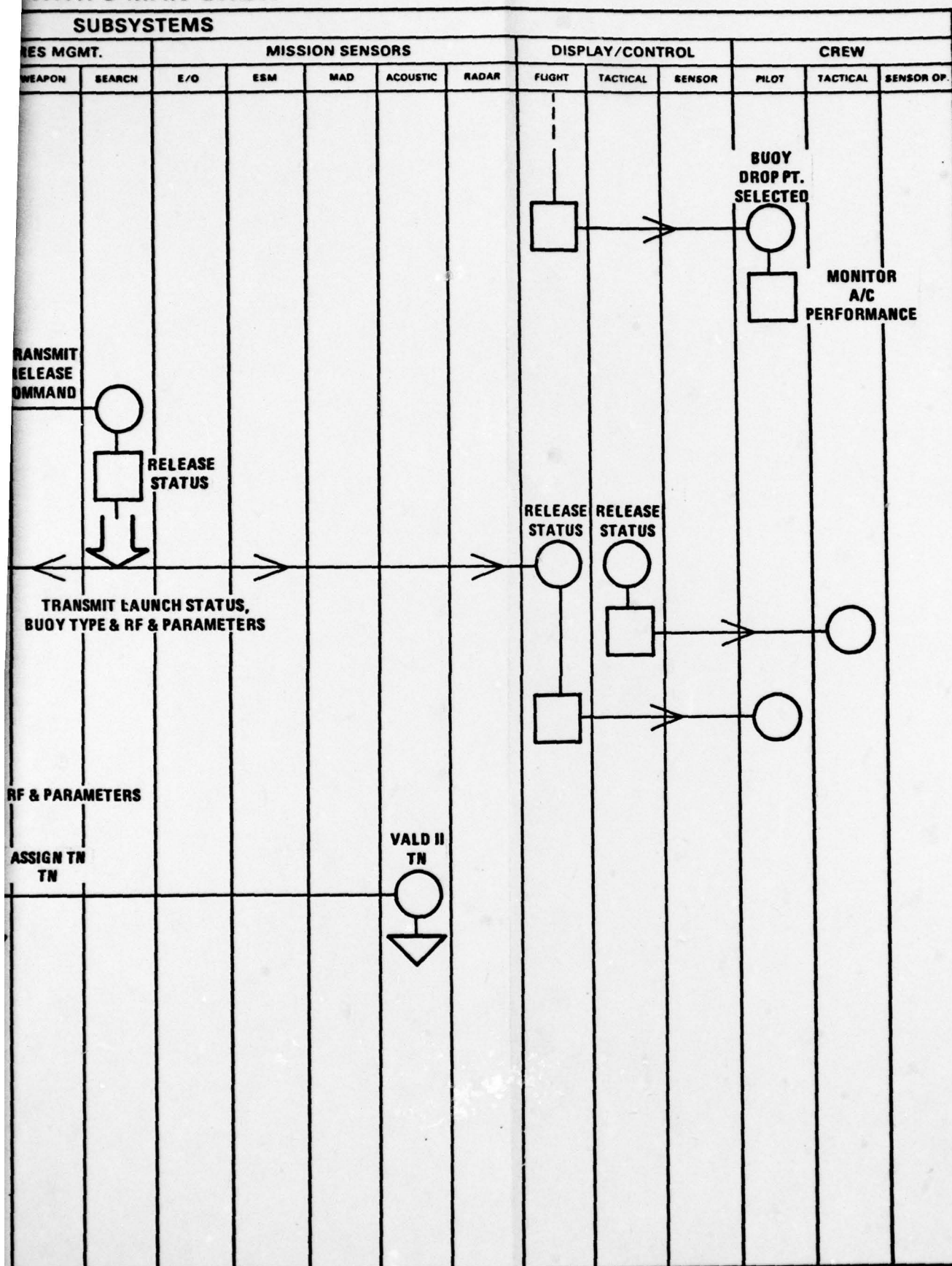


# VSTOL F3 AVIONICS CONCEPT WITH



**WITH 3 MAN CREW**

## SUBSYSTEMS

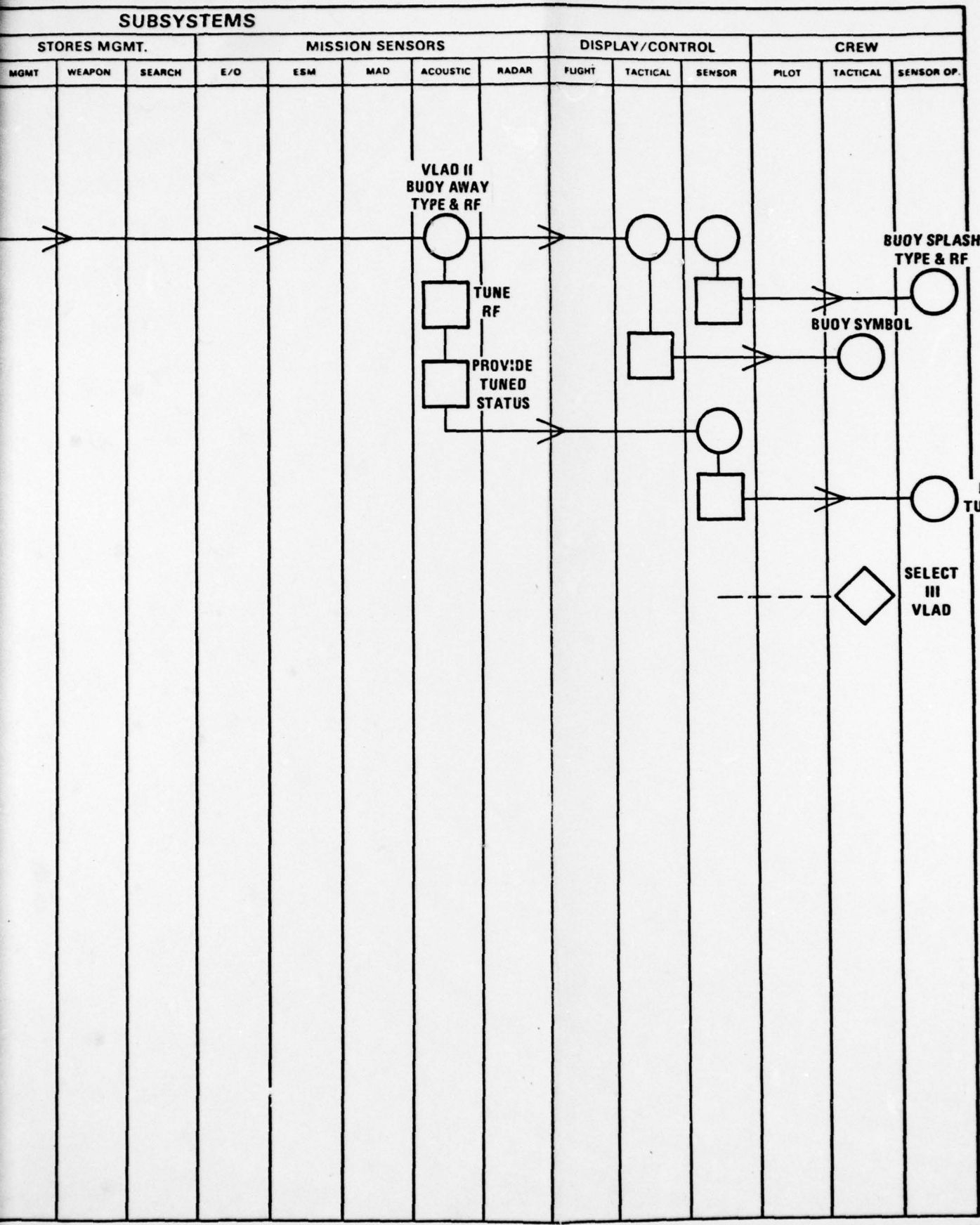






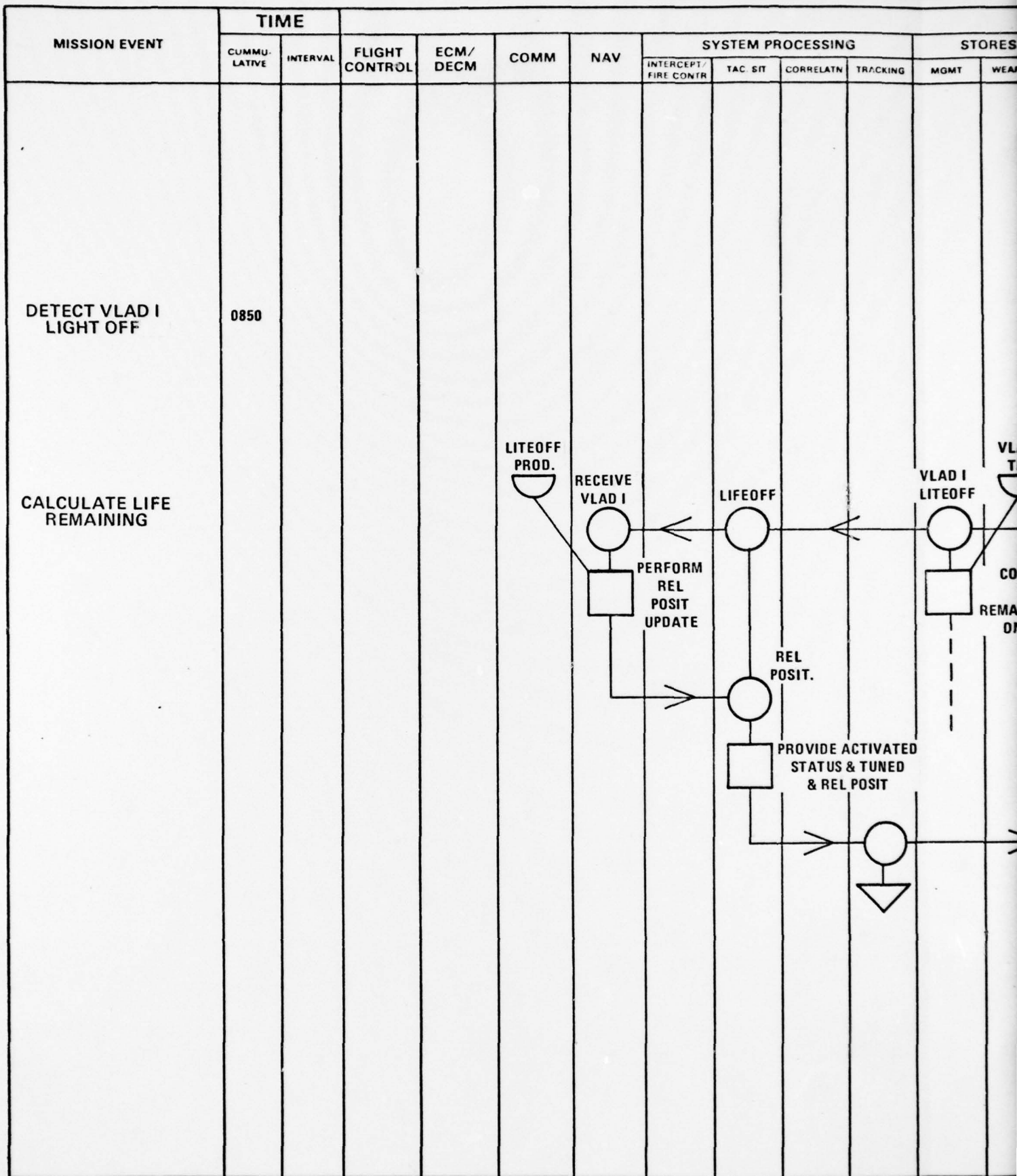


CONCEPT WITH 3 MAN CREW



2

# VSTOL F3 AVIONICS CONCEPT W



## SUBSYSTEMS



# DISTRIBUTION LIST

REPORT NO. NADC-79161-40

AIRTASK NO. 672IN/F21200/WF21200000  
Work Unit No. ZC909

	<u>No. of Copies</u>
NAVAIRSYSCOM, AIR-950D . . . . .	7
(2 for retention)	
(1 for AIR-503)	
(1 for AIR-533)	
(3 for AIR-360B - Mr. Zempolich)	
CNO, OP-03G . . . . .	2
COMOPTEVFOR . . . . .	1
NAVELECSYSCOM . . . . .	1
NAVSHIPSYSYSCOM, Code 2051 . . . . .	1
AIRTEVRON ONE . . . . .	1
NELCEN . . . . .	1
NUSC, New London . . . . .	1
NUSC, Newport . . . . .	1
NSRDC . . . . .	1
NAVWPNSCEN . . . . .	1
NURDC . . . . .	1
NRL . . . . .	1
NAVSURFWPNCEN . . . . .	1
NAVAIRTESTCEN . . . . .	1
NAVPGSCOL . . . . .	1
NAVIONICFAC . . . . .	1
NAVMISCEN . . . . .	1
AFSC (SCTS), Andrews AFB . . . . .	1
ERADCOM (DELSO-L) . . . . .	1
DDC . . . . .	12